Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería de Telecomunicación

IMPLEMENTACIÓN DE MECANISMOS AVANZADOS DE MOVILIDAD DE REDES SOBRE ROUTERS FONERA

PROYECTO FIN DE CARRERA

Autor: Jose Pablo Salvador Garcia
Tutor: Carlos Jesus Bernardos Cano

Octubre de 2010

Proyecto Fin de Carrera
IMPLEMENTACIÓN DE MECANISMOS AVANZADOS DE MOVILIDAD DE REDES SOBRE ROUTERS FONERA

Autor
José Pablo Salvador García

Tutor
Dr. Carlos Jesús Bernardos Cano

La defensa del presente Proyecto Fin de Carrera se realizó el da 28 de Octubre de 2010, siendo calificada por el siguiente tribunal:

Presidente: Antonio de la Oliva Delgado

Secretario: Julio Villena Román

Vocal: Jerónimo Arenas García

y habiendo obtenido la siguiente calificación:

Calificación: 10

Leganés, a 28 de Octubre de 2010

A mis padres
A mi hermana
A Isabel

Agradecimientos

En primer lugar, se lo quería agradecer a Isabel, sin ella todo esto no hubiese sido posible. Ella es mi guía, mi soporte, mi compañera y mi amiga. Gracias a ella hoy estoy aquí presentando mi proyecto.

A continuación y, por supuesto, en igual nivel de importancia, a mis padres y mi hermana. Gracias a vosotros soy lo que soy, para lo bueno y para lo malo. Espero que estéis muy orgullosos de mi, como yo lo estoy de vosotros. A Ana, gracias a ti me costaba menos arrancar por las mañanas.

Como no agradecerle a mi tutor, Carlos, por su dedicación, su tiempo y su paciencia. Ha sido un verdadero placer trabajar con él y me encantaría seguir haciéndolo en un futuro. Para mí es un ingeniero e investigador excepcional. Siempre se aprende algo de él cada día. También a María Calderón, gracias a ella empezó mi colaboración en este departamento. Su cercanía y su paciencia han hecho mucho más fácil mi colaboración e investigación en un mundo nuevo para mí.

Tampoco me puedo olvidar de grandes personas y amigos que he conocido durante la carrera. Sin ellos tampoco estaría aquí. Hemos pasado lo indecible para llegar al final, pero se llega. Hemos disfrutado, reído, llorado, peleado, para mí habéis sido a veces mi familia. Me gustaría destacar a: Ra, MeryLoki, Little D, Boto, Carol, Chalao, Martos, Mertxis, Joak y Cristo. Sólo puedo decir que sois increíbles y muchísimas gracias por ser mis amigos. Otras grandes personas que han formado parte de todo esto, aunque sea en menor medida, pero muy importantes en mi corazón son: Hex, María S.B., Cris y Nieves.

Por último, me gustaría destacar esta última etapa de mi vida en Alemania. Allí he conocido a gente increíble (Stefano, Marco, María y Patricia) que en momentos de flaqueza estuvieron ahí para apoyarme y levantarme el ánimo.

«L'amicizia è tutto. L'amicizia è più del talento. È più del governo. È quasi uguale alla famiglia. Non dimenticarlo mai».
Vito Corleone a Johnny Fontane

No digas: es imposible. Di: no lo he hecho todavía.
Proverbio japonés

Resumen

Al hablar de telecomunicaciones, el primer pensamiento de cualquier persona suele ser acerca de Internet, un ordenador o un teléfono móvil. En los últimos tiempos se hace cada vez más habitual encontrar a gente utilizando su portátil o su PDA para acceder a su correo electrónico, mantener una reunión a larga distancia o realizar otro tipo de gestiones. Tampoco es algo extraño que esto ocurra en una estación de tren, en un aeropuerto o un centro comercial. Pero, ¿qué ocurriría si se pudiese dar acceso a Internet en escenarios tan diversos y además, en movimiento? Los protocolos de movilidad ya hacen posible que exista conexión a Internet en un medio de transporte, por ejemplo, pero si un usuario quiere utilizarlo debe tener un dispositivo que soporte alguno de los protocolos de movilidad, y al mismo tiempo, si todos los usuarios quisieran utilizarlo a la vez, supondría una sobrecarga de información de control entre cada usuario y la red.

Con el fin de mejorar los protocolos de movilidad existentes y extenderlos de forma que se soporte la movilidad de redes completas, surgió el protocolo NEMO BS, *NEtwork MObility Basic Support* o Protocolo de Soporte Básico de Movilidad de Redes (RFC 3963 [DWPT05]). Este protocolo realiza una extensión de MIPv6, *Mobile Internet Protocol v6*, RFC 3775 [JPA04], definiendo un nuevo dispositivo, el *router* móvil, que centraliza la gestión de la movilidad de la red, liberando de esta carga a cada uno de los nodos en la red móvil.

En este proyecto fin de carrera se ha desarrollado una implementación de este protocolo para un tipo de *routers* de bajo coste y reducido tamaño, llamados Fonera, que actuarán como *routers* móviles, bajo una distribución de Linux adaptada específicamente para ellos. Además, se han utilizado otros dispositivos, tales como *routers* Linksys, ordenadores portátiles, etc. para desarrollar un escenario en el que existan los distintos tipos de entidades involucradas en el protocolo, desarrollando una implementación a partir de una ya existente, extendiéndola y complementándola con otros mecanismos que se expondrán a lo largo de esta memoria.

En los últimos tiempos, la investigación en redes vehiculares ha ido adquiriendo mayor importancia con el objetivo de aumentar la seguridad del tráfico en las carreteras. Sin embargo, hoy en día, aparecen nuevas opciones, basadas en el uso de Internet para aplicaciones de entretenimiento. El protocolo NEMO BS puede aplicarse a este tipo de redes de distintas formas, obteniendo diferente impacto de acuerdo a términos económicos, funcionales y de rendimiento.

Con la motivación de simular un escenario de comunicaciones móviles se ha desarrollado una plataforma de demostración, formada por coches radio control, RC, cámaras IPv6 y routers Fonera que posteriormente es utilizada en el proyecto español POSEIDON.

Palabras clave: IPv6, Movilidad de Redes, Router Móvil, Redes Vehiculares.

Abstract

When talking about telecommunications, anyone's first thought is focused on the Internet, a computer or a mobile phone. Nowadays it is becoming more and more common to see people using their laptop or PDA in order to access their mail, having a videoconference or performing another kind of activities. It is not so weird either to see those scenes happening in a train station, an airport or a shopping centre. But, what would happen if anyone could access the Internet in those diverse scenarios and also in movement? Mobility protocols make already possible Internet connections, for example in a transport system, but if users must have an adequate device supporting mobility protocols and, at the same time, if everyone wanted to use it simultaneously, that could provoke an overload of control information between each user and the network.

With the aim of improving the existing mobility protocols and extend them to provide support for the movement of complete networks, the NEMO BS protocol was developed - *NEtwork MObility Basic Support* - (defined in RFC 3963 [DWPT05]). This protocol extends MIPv6, defining a new entity, the mobile router, which performs all the management tasks for the mobility of the network, on behalf of the other devices.

In this Master Thesis it has been developed an implementation of this protocol using COTS (*Commercial Off-The-Shelf*) devices, particularly FON routers that will act as mobile routers, under a Linux distribution especially adapted for them. Moreover, other devices have been used, such as Linksys routers, laptops, PC's, and so on, in order to configure a scenario where all the entities involved in the protocol can exist, developing an implementation from a previous existing one, expanding it and adding new features that will be presented along this document.

Research in Vehicular Ad-hoc Networks (VANETs) has emerged in such a great way past decade. The original motivation of these vehicular networks was to increase the safety in lanes. However nowadays, infotainment[1] applications are getting more and more popular. NEMO protocol can apply to these kind of networks in different ways, obtaining different impacts in terms of economic, functional and performance requirements.

In order to simulate a vehicular network a testbed has been deployed which is composed of remote control cars, IP cameras and Fonera routers. Later on, this testbed has been integrated into the Spanish Project POSEIDON.

Keywords: IPv6, Network Mobility, Mobile Router, Vehicular Networks.

[1] Information and entertainment applications. Information-based media content or programming that also includes entertainment content in an effort to enhance popularity with audiences and consumers.

Indice General

Lista de Figuras

Lista de Tablas

Capítulo 1

Planteamiento y objetivos

1.1. Introducción

Este documento describe el trabajo realizado como proyecto fin de carrera, consistente en el estudio del protocolo de Soporte Básico de Movilidad de Redes (*Network Mobility Basic Support Protocol*, NEMO BS) - especificado en la RFC 3963 [DWPT05] - y su correspondiente implementación práctica.

Este protocolo aporta una solución a la problemática de la movilidad IP, ya que el protocolo de nivel de red IP (*Internet Protocol*) no fue diseñado teniendo en cuenta la posibilidad de que los dispositivos que forman una red o incluso la red completa se movieran.

Una primera solución a este problema de la movilidad fue el protocolo MIPv6 (*Mobile IPv6*), que permite el movimiento de un terminal cambiando su punto de acceso a Internet manteniendo la conexión. NEMO BS es una solución que amplía la anterior, posibilitando el movimiento de redes completas, no de un único nodo. Además la red no tiene por qué estar formada por el mismo tipo de dispositivos, ni siquiera tienen que ser ordenadores, pueden ser PDA's (*Personal Digital Assistant*), teléfonos móviles, cámaras, etc.

NEMO introduce un nuevo elemento, el router móvil, que es el principal cambio con respecto al protocolo MIPv6. Este router va a ser el encargado de dar soporte a la movilidad de la red, liberando de complejidad adicional al resto de terminales finales o dispositivos que conforman dicha red.

El trabajo de este proyecto se centra en desarrollar una aplicación o un *software* para implementar las funciones del router móvil en un dispositivo de bajo coste, el router Fonera 2200. Por otro lado, también se realiza otra aplicación que implementa las funciones de la otra entidad implicada en el protocolo, en este caso, el agente local (*Home Agent*).

Existen diversas tecnologías que permiten la conexión a Internet de dispositivos inalámbricos, inherente al tema de movilidad de redes, de forma continua permitiendo movilidad completa, tales como UMTS[1] o HSDPA[2]. Estas tecnologías comparadas con WiFi [3] presentan varios inconvenientes que hacen que queden descartadas: ambas conllevan un elevado coste, y además en el caso de UMTS un peor rendimiento (ancho de banda,

[1] UMTS: ***U**niversal **M**obile **T**elephone **S**ystem*. Tecnología 3G.
[2] HSPA: ***H**igh **S**peed **P**acket **A**ccess*. Tecnología 3G+
[3] WiFi: Marca del estándar IEEE 802.11.

retardo).

Además, WiFi, cuyo estándar es el 802.11 [iee07], es una de las tecnologías más comúnmente utilizadas y facilita la compatibilidad con un mayor número de dispositivos. Por estas razones, en este proyecto se emplea dicha tecnología. Otra tecnología que podía haberse considerado es WiMAX[4] cuyo estándar es el 802.16. Esta tecnología permite un alcance mucho mayor que WiFi, distancias de 50-60 km frente a los 100 metros de WiFi, sin embargo el desarrollo y despliegue de esta tecnología aún no es completo. Además su largo alcance obligarían a simular los cambios de punto de acceso, en lugar de poder probarlos en un escenario real, debido a las limitaciones de espacio existentes en el laboratorio.

Un ejemplo claro del uso de este protocolo puede darse en transportes públicos, como se muestra en la Figura 1.1.

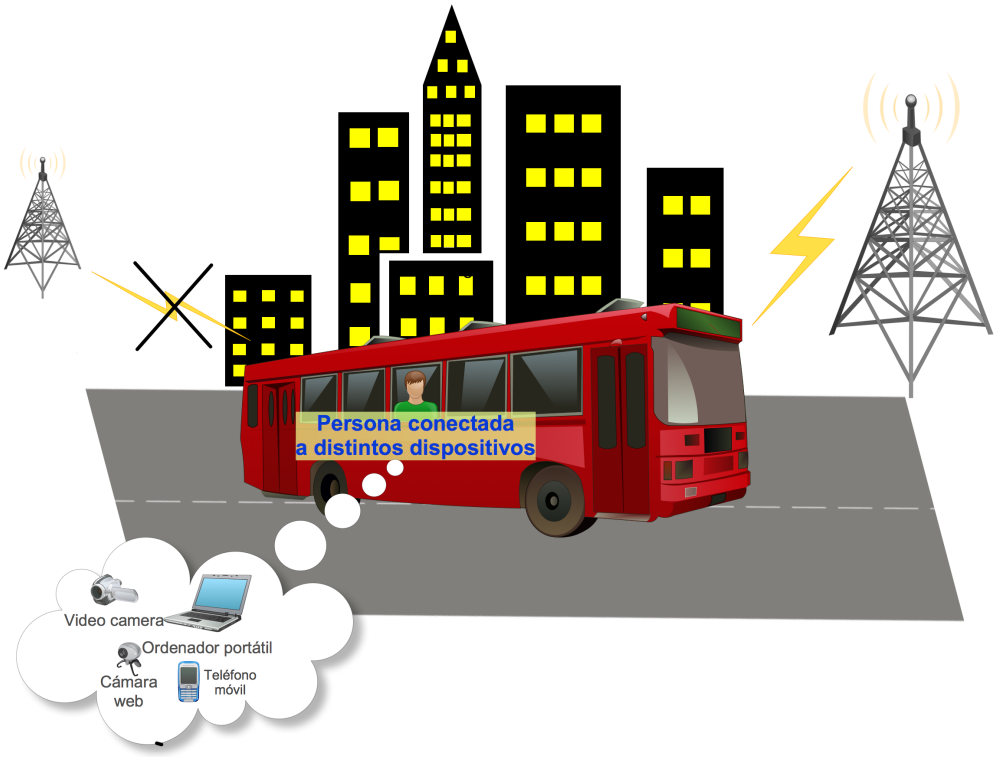

Figura 1.1: Acceso a Internet en transportes públicos.

[4]WiMAX: ***W****orldwide* ***I****nteroperability for* ***M****icrowave* ***A****ccess*

1.2. Objetivos

Los principales objetivos de este proyecto se enumeran a continuación:

- Estudio del protocolo de Soporte Básico de Movilidad de Redes propuesto por el IETF (*Internet Engineering Task Force*)[5] a través de la asimilación del documento donde se recoge su especificación [DWPT05].
- Análisis del *firmware* libre OpenWrt y de los procesos de cambio de *firmware* en los distintos tipos de routers utilizados.
- Implementación de una solución que cubra la funcionalidad básica del protocolo NEMO BS, tanto para un router móvil como para un agente local.
- Extender la funcionalidad a partir de una implementación anterior para que el router móvil dé soporte a varios terminales y el agente local también soporte a varios router móviles.
- Despliegue y configuración de un escenario de red, lo más aproximado a un escenario real. Esta plataforma de desarrollo es utilizada en el proyecto español POSEIDON - ***P**rovisión **Ó**ptima de **SE**rvIcios a re**D**es vehiculares en m**O**vimieNto* (TSI2006-12507-C03).
- Evaluación práctica de la solución llevada a cabo.
- Análisis del rendimiento y las prestaciones para la validación de la solución implementada.

1.3. Estructura de la memoria

La presente memoria se divide en los siguientes capítulos:

- **Capítulo 1. Planteamientos y Objetivos.** En este primer capítulo se hace una breve introducción de las causas y motivaciones que originan la realización de este proyecto. También se presentan los objetivos y la estructura del documento.
- **Capítulo 2. Movilidad de redes IP: NEMO.** Se analiza el estado del arte de la movilidad de redes, así como las soluciones existentes para gestionar dicha movilidad.
- **Capítulo 3. Desarrollo de un prototipo de NEMO BS.** En este capítulo se detallan de forma exhaustiva los pasos seguidos en el desarrollo de la aplicación realizada y las decisiones de diseño adoptadas.
- **Capítulo 4. Evaluación del prototipo.** En este capítulo se describe el escenario configurado para la evaluación del prototipo, presentando a continuación las pruebas de funcionamiento y rendimiento realizadas a dicha implementación.
- **Capítulo 5. Conclusiones y futuras líneas de trabajo.** En este capítulo se presentarán las conclusiones más importantes que se pueden extraer del trabajo realizado, así como posteriores líneas de trabajo.

[5] http://www.ietf.org

- **Apéndice A. Presupuesto (planificación de tareas, costes, etc.).** Se presenta un análisis de las tareas realizadas y los costes derivados de este proyecto.
- **Apéndice B. Mensajes del protocolo de Soporte Básico de Movilidad de Redes - NEMO BS.** En este apéndice se detalla la estructura de la cabecera de movilidad y de los mensajes de señalización del protocolo NEMO BS.
- **Apéndice C. Instalación de OpenWrt en la Fonera.** Estudio del proceso de cambio de *firmware* en el router Fonera 2200.
- **Apéndice D. Instalación de OpenWrt en el Linksys WRT54G.** Estudio del proceso de cambio de *firmware* en el router Linksys WRT54G.
- **Apéndice E. Instalación y uso de la plataforma de desarrollo de OpenWrt en PC.** Estudio de las herramientas de desarrollo que proporciona OpenWrt.
- **Apéndice F. Montaje y configuración del escenario desplegado.** Descripción del proceso de montaje y configuración del escenario en el que se realiza la evaluación experimental del prototipo.
- **Apéndice G. Instalación y configuración del *software* desarrollado.** Manual de usuario de la implementación realizada.
- **Apéndice H. *Streaming* de vídeo.** Estudio de las distintas posibilidades de *streaming* de vídeo compatibles con las características de la aplicación.
- **Apéndice I. *Hardware* adicional.** Descripción del *hardware* adicional para dotar de autonomía a los dispositivos de la red móvil, y pruebas de evaluación.

Capítulo 2

Movilidad de redes IP: NEMO

2.1. Introducción

En este capítulo se presentan de forma breve los principales antecedentes en los que se basa la realización de este proyecto. En primer lugar, se introduce el concepto de movilidad para analizar con particular interés la problemática de la movilidad de redes IP, dando pie a la necesidad de una solución como el protocolo NEMO *Basic Support*, también conocido como NEMO BS.

Por otra parte, se explicará en que consiste esta solución y los cambios introducidos respecto al protocolo MIPv6, en el cual se basa, extendiendo sus funcionalidades.

2.2. Movilidad

En los últimos años ha habido un gran desarrollo tanto de las tecnologías inalámbricas - GSM, GPRS, UMTS, HSPA, Bluetooth, WiFi, WiMAX, Mobile WiMAX - como de los dispositivos móviles, que cada día son capaces de incluir mayor número de funcionalidades con un tamaño menor. Este desarrollo favorece la aparición de nuevas aplicaciones, algunas muy atractivas para los usuarios. Al mismo tiempo los propios usuarios cada vez demandan más la posibilidad de tener acceso a Internet, o a otras redes - red personal o de trabajo -, prácticamente desde cualquier sitio y de forma inalámbrica, conformando un sistema complejo y heterogéneo, como por ejemplo, el esquema mostrado en la Figura 2.1. Particularmente llamativo es el aumento de la necesidad de acceso a Internet desde elementos en movimiento, como por ejemplo los medios de transporte.

Antes de entrar en más detalles, es importante aclarar lo que significa la movilidad y los problemas que hay que enfrentar para dar soporte de movilidad en las redes existentes. Comentar que la movilidad no es un servicio en sí, no tiene valor si no va acompañado de otros servicios. Lo que hace es mejorar su disponibilidad. A menudo, se confunde con la portabilidad, que es la capacidad de tener acceso a Internet en cualquier localización, normalmente de forma inalámbrica. Sin embargo, es necesario configurar esa conexión, que será cerrada durante el cambio de una red a otra. Por otra parte, la movilidad facilita el acceso de forma transparente al usuario, sin necesidad de reconfiguración, permitiendo cambiar de punto de acceso manteniendo la conexión incluso en movimiento. Si se provee de movilidad, la portabilidad también será soportada, pero no al revés. La movilidad para

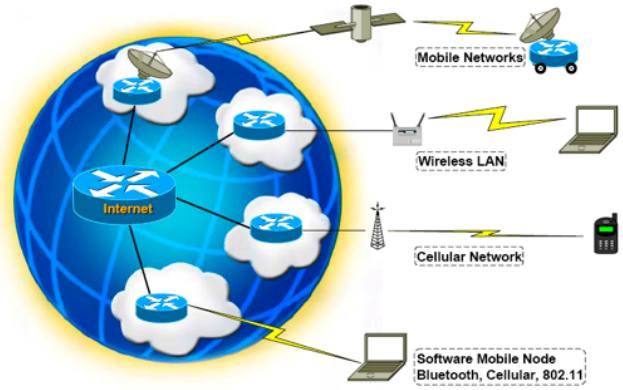

Figura 2.1: Global Internet [Cisco].

ser considerada como tal debe cumplir unos **requisitos**:

- **Continuidad en la sesión**. Es necesario que cuando un dispositivo cambie de red todas sus sesiones y conexiones abiertas no sufran cortes y no sea necesario reiniciarlas.

- **Accesibilidad en Internet**. Es crucial para los usuarios seguir siendo alcanzables en Internet a través de su dirección IP. No debe tener limitaciones geográficas. El terminal tiene que trabajar en cualquier parte.

- **Independencia de la aplicación**. Un mecanismo de movilidad no puede estar pensado para una aplicación en concreto, sino que debe dar soporte a cualquier aplicación, sin que sea necesario realizar ningún cambio en ella. Es decir, debe hacer transparente a las capas de transporte y aplicación cualquier cambio de red. Se debe poder comunicar con cualquier nodo de red, aunque no implementen movilidad, sin modificar el direccionamiento estándar.

- **Independencia de la tecnología de red**. Se conecta automáticamente en cualquier localización. No parece razonable que una implementación de movilidad ofrezca restricciones en cuanto a las tecnologías de las capas de enlace y física. Por esta razón, entre otras, parece sensato plantear un protocolo basado en IP(v6) como la mejor opción para proporcionar movilidad, ya que este protocolo puede ser usado con cualquier tecnología de acceso.

- **Señalización extremo a extremo**. Es necesario que la solución no implique ningún cambio en los routers intermedios y toda la inteligencia de la implementación resida en nodos o routers finales. Así se evitarían problemas de escalabilidad.

- **Seguridad**. Debe realizar las funciones de movilidad en cualquier localización y poder intercambiar información protegida.

2.2.1. Problemática en torno a la movilidad

Para proveer de movilidad a los usuarios surgen varios problemas, para los que es necesario encontrar una solución:

- Las redes IP no fueron pensadas para soportar cambios en el punto de acceso a la red. Si se configura una red es posible hacerla funcionar, ofrecer distintos parámetros de QoS y opciones de diferente complejidad para mejorar o adaptar a las necesidades particulares de cada cliente las características de la red, pero en el momento en el que se intenta cambiar de punto de acceso a la red y además de forma dinámica y transparente las cosas se complican.

- Las direcciones IP cumplen dos papeles:

  1. Por un lado funcionan como localizadores, indicando mediante mecanismos de encaminamiento como llegar al terminal al que identifican. Este sistema de encaminamiento mantiene información de cómo llegar a conjuntos de direcciones que comparten un prefijo de red. Es necesario mantener todas las direcciones IP de la red topológicamente correctas (coherencia de prefijos de red).
  2. Por otro lado, las direcciones IP actúan como parte de los identificadores de los puntos extremos de una comunicación, utilizados por los niveles superiores para identificar la comunicación (Identificación conexiones TCP/UDP por el conjunto dirección IP:puerto).

2.2.2. Soluciones para la movilidad

Las posibles soluciones que se pueden plantear serían:

- Cuando un terminal salga de su red habitual, identificada por el prefijo de su dirección IP, reconfigurar todos los routers involucrados actualizando en las tablas de rutas la nueva posición. Esto presenta varios problemas, como la escalabilidad, el alto retardo que introduciría y la vulnerabilidad en temas de seguridad.

- La otra opción sería cambiar la dirección IP del dispositivo móvil para mantener la corrección topológica en la nueva red, pero esto conllevaría a una implementación de portabilidad, no de movilidad, ya que impondría un cambio en la conexión que necesita la reconfiguración de parámetros y reiniciar la sesión con una nueva dirección IP, que impediría que el terminal se moviera de una red a otra manteniendo la conexión.

Teniendo en cuenta estas consideraciones, se introduce MIP (*Mobile IP*), una extensión de IP con el objetivo de proporcionar movilidad a los usuarios, es decir, darles la capacidad de cambiar de punto de acceso mientras mantienen sus conexiones. Es una solución a nivel de red, que parece la mejor alternativa para solventar un problema que reside principalmente en cambios de direcciones IP. Además, desde el punto de vista estructural, IP implica una única solución, ya que es soportado por la inmensa mayoría de protocolos de capas tanto inferiores como superiores. Sin embargo, dar soporte de movilidad a nivel IP quizá no sea la mejor opción para proveer adaptación a la aplicación,aunque sí es lo más acertado para gestionar los cambios de rutas y el rendimiento.

El IETF ya estudió el desarrollo de capas de red para permitir la movilidad de terminales en redes IP, como es el caso de MIPv4 [Per02] para IPv4 y MIPv6 [JPA04] para IPv6, pero estos protocolos no soportan el movimiento de una red completa. Ahí es donde surge NEMO, de NEtwork MObility, definido en la RFC 3963 [DWPT05], para extrapolar esa movilidad de los terminales a una red que se mueve como un todo, cambiando

su punto de acceso a la infraestructura de red fija. Además, este movimiento debe resultar transparente a cada uno de los nodos de la red, siendo un único elemento el que soporta la complejidad de gestionar el movimiento de la red completa. Este elemento como se verá con más detalle a continuación es el router móvil.

No es solamente el hecho de que no existiese un protocolo que hiciese posible o contemplase la movilidad de una red completa lo que crea la necesidad de que aparezca NEMO, sino la cantidad de aplicaciones que podría tener la movilidad de redes. La Figura 2.2 muestra varios escenarios en los que se pueden encontrar redes inalámbricas, o donde al menos, puede ser más aconsejable este tipo de red que una cableada, por distintas razones: accesibilidad, variedad de usuarios, facilidad de configuración, o simplemente la comodidad de disfrutar de un acceso de banda ancha sin la necesidad de estar conectado a un cable. Como complemento a una red inalámbrica, el número de aplicaciones crecería significativamente si estas soluciones pudieran proporcionar movilidad a sus usuarios.

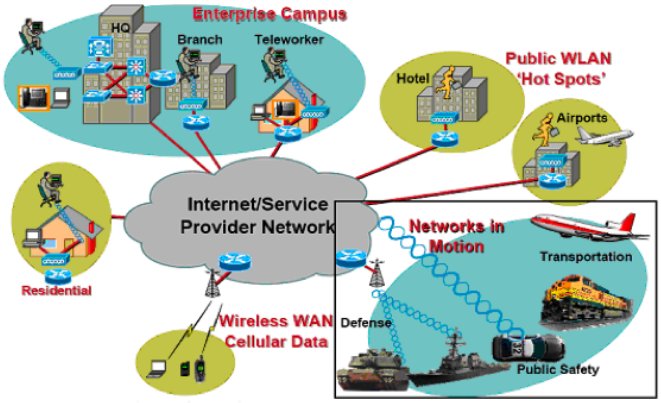

Figura 2.2: Escenarios posibles para *Mobile IP* [Cisco].

Por ejemplo, algunas de las más interesantes serían proporcionar acceso a Internet a distintos tipos de redes:

- La primera opción en que se suele pensar, es en el acceso a Internet en el transporte público.
- Redes de área personal (PAN). Distintos dispositivos electrónicos que una persona puede llevar encima (PDA, cámara de fotos, reproductor de música, etc.) que obtienen acceso a Internet a través de, por ejemplo, un teléfono móvil que actúa como MR.
- Redes vehiculares. Al igual que en el transporte público, puede ser interesante dotar de acceso a Internet a los pasajeros que van en otra clase de vehículo, incluso dando origen a otro tipo de servicios, como permitir que hoteles, estaciones de servicio u otros elementos de interés elegidos por el usuario se "anuncien" al viajero en un radio de x kilómetros.
- Redes *ad hoc* en general. Redes inalámbricas en la que los equipos pueden comunicarse entre sí sin necesidad de un punto de acceso.
- Redes de sensores desplegados en un vehículo, que se comunican con el exterior a través de un router móvil.

2.3. Desarrollo de NEMO BS

Como se comenta en la anterior sección el protocolo NEMO[1] surgió como un grupo de trabajo del IETF con la intención de solucionar los problemas derivados de la movilidad de redes, tales como la gestión de la movilidad de una red completa en la que se producen, de forma conjunta, cambios de punto de acceso en Internet y por tanto de su situación en la topología de Internet.

Antes de pasar a comentar con mayor detalle NEMO BS, se van a identificar los puntos débiles de MIPv6 y su origen para ver los motivos que llevaron a crear dicho grupo de trabajo. En primer lugar, el protocolo MIP introdujo una serie de entidades funcionales para dar soporte de movilidad de forma que el nodo móvil continuase siendo identificado por su dirección IP, sin importar donde se conectase a Internet, y además, pudiera moverse cambiando su punto de acceso ya que el protocolo soporta la itinerancia - *roaming* -, aunque el tiempo de *handover* o traspaso puede causar algunas ineficiencias.

Dado que se creó como extensión de IPv4, MIPv4 tiene varios puntos débiles. Alguno de ellos son mejorados o resueltos en la especificación de MIPv6, que se beneficia de la experiencia adquirida con MIPv4 y las novedades que proporciona IPv6, que fue diseñado teniendo en cuenta la integración de la movilidad, característica lejos de ser considerada cuando se diseñó IPv4.

Teniendo en cuenta la variedad de aplicaciones de estas extensiones de IP, es natural que existan diferentes grupos de trabajo y entidades dedicados a la investigación en la mejora de su rendimiento, tratando de solventar sus inconvenientes.

- Por ejemplo, el grupo MIPSHOP - *MIP Signalling and Handoff Optimization* - que trabaja en soluciones de micromovilidad, como HMIP -*Hierarchical MIP* - [SCMB05], implementación jerárquica de MIP basada en una estructura de árbol. Introduce mayor complejidad pero disminuye la latencia. HMIPv6, construido sobre MIPv6, diferencia entre movilidad local y global: la movilidad global es controlada por MIPv6 mientras que los *handoffs* locales son gestionados de forma local, gracias a un nuevo nodo, MAP (*Mobility Anchor Point*), que disminuye el retardo ya que puede ser actualizado más rápidamente que un agente local (HA) remoto.
- Otra posibilidad para proporcionar micromovilidad es *Hawaii* [Ram99], que establece esquemas especiales de enrutado en nodos específicos del dominio. En lo referente a la mejora del tiempo de *handoff* o *handover* (traspaso a otra red cuando el nivel de señal en el enlace se considera insuficiente), también se han estudiado soluciones dentro del grupo MIPSHOP para conseguir un *handover* rápido - *fast handover* - para MIPv6 [Koo05].
- Otra mejora proporcionada a MIPv6 es MIPv6 *Bootstrapping*, que permite eliminar la configuración de forma estática de la dirección del HA y la HoA en los nodos móviles.
- Otro aspecto relevante a tener en cuenta en los entornos de movilidad es la seguridad. A pesar de que por el hecho de utilizar IPsec [ADD04], MIPv6 es más seguro que MIPv4, existen algunas líneas de trabajo dirigidas a mejorar este punto, sobre todo en lo referente a direcciones IPv6 [Aur05], ya que es importante asegurar la privacidad y el anonimato de los nodos móviles.

[1] http://datatracker.ietf.org/wg/nemo/charter/

- Finalmente, también cabe mencionar que existen algunas soluciones específicas para incluir la movilidad en distintos escenarios y combinarla con otras redes existentes. Es el caso de MANET - *Mobile Ad hoc Networks* - que trabaja en la interconexión de redes *ad hoc* y redes móviles [Zul]. Comentar que las redes MANET no tienen por qué tener soporte de movilidad a nivel IP, aunque algunas soluciones lo propongan.

2.3.1. Comparativa MIPv6 *vs* NEMO BS

Centrando el debate en redes IPv6, MIPv6 no soporta el movimiento de una red completa. Se podría conseguir movilidad de forma transparente para todos los nodos de una red que se mueven juntos, habilitando soporte de movilidad en cada uno de ellos, para que gestionaran de forma individual su movilidad. Sin embargo, existen claras desventajas:

- Sería necesario que cada uno de los nodos de la red tuviera soporte de movilidad. En determinadas redes esto puede no ser posible, debido a los recursos limitados de algún nodo (redes de sensores o dispositivos empotrados) o porque determinado *software* sea incompatible con sistemas antiguos.
- La señalización necesaria en ese caso causaría una sobrecarga en la red debido a la cantidad de mensajes que enviaría cada nodo de la red, que en determinadas ocasiones puede ser un número considerable.

Así, en el desarrollo de NEMO, se trataba de modificar la solución aportada por MIPv6 para la movilidad de nodos para proporcionar movilidad a una red completa. Además, esta opción debía tener cierta flexibilidad, para ser compatible con las posibles configuraciones distintas de la red móvil. Las desventajas anteriores se solventan gracias a la presencia de un nuevo nodo, el router móvil, que gestiona la movilidad de la red completa, sin que los nodos que la forman necesiten implementar ningún *software* específico para disfrutar de movilidad de forma transparente. Además, el proceso de señalización debido a la itinerancia de la red se limita a un único nodo enviando un sólo mensaje. Otra ventaja de la presencia del MR consiste en que este nodo proporciona acceso a Internet, de forma que no es necesario que cada uno de los nodos implemente una tecnología específica para la movilidad, sino que se podría utilizar una tecnología más barata y/o comúnmente utilizada, liberando de complejidad o coste a cada nodo de la red.

A continuación se va a explicar con más detalle el funcionamiento del protocolo. Sin embargo, para un mejor entendimiento del mismo, se van a describir de forma breve las principales entidades que componen la topología de la red que se va a emplear.

2.3.2. Terminología

En este apartado se van a explicar las distintas entidades que se pueden encontrar en una red móvil y que tienen relevancia para poder entender posteriormente el funcionamiento del protocolo NEMO.

A continuación se van a enumerar y explicar otros conceptos importantes para el correcto entendimiento del protocolo:

- Prefijo de la red móvil o *Mobile Prefix Node* (MNP). Prefijo perteneciente a la red móvil o al MR. Cada red móvil puede tener uno o varios prefijos.

- Interfaz externa o *egress interface*. Es la interfaz que conecta al MR con la red de acceso que está utilizando.
- Interfaz interna o *ingress interface*. Es la interfaz a la que se conectan todos los equipos de la red móvil.
- Dirección Hogar o *Home Address* (HoA). Es la dirección IPv6 perteneciente a la interfaz interna, aunque dependiendo de la implementación también se puede considerar como la dirección que toma el MR cuando regresa a la red hogar. Pertenece al espacio de direcciones de la subred hogar del router móvil, que puede o no coincidir con el MNP.
- Dirección temporal en la red visitada o *Care-of Address* (CoA). Dirección IPv6 con la se configura la interfaz externa cuando el router móvil se encuentra en una red visitada. La forma en la que el MR configura esta dirección es a partir del anuncio de router o *router advertisement* que envían los routers de acceso junto con la dirección MAC de la interfaz externa del MR (formato de dirección EUI-64).

Ahora se van a comentar las principales entidades a nivel de red que se encuentran en el escenario de NEMO:

- En primer lugar destaca, el router móvil o *mobile router* (MR), que es la nueva entidad que se introduce en NEMO BS con respecto a protocolos previos. Como se menciona en la subsección 2.3.1, este dispositivo gestiona la movilidad de la red completa, permitiendo que el resto de sus nodos sean alcanzables permanentemente a través de su red hogar. Además, es el encargado de detectar el movimiento de la red móvil.
- Agente local o *home agent* (HA). Es un router o equipo en la red hogar del router móvil, en el que éste registra su HoA y CoA. Tiene como misión recibir los paquetes enviados a cualquier nodo de la red móvil cuando ésta no se encuentra en su red hogar y reenviárselos al MR a través de un túnel que tiene como dirección destino la CoA del MR.
- Router de acceso o *access router* (AR). Routers externos a los que se conecta la red móvil cuando está fuera de la red hogar.
- Nodo corresponsal o *correspondent node* (CN). Nodo o equipo que se comunica con un nodo de la red móvil.

2.3.3. Funcionamiento del protocolo

Como ya se ha comentado, el router móvil es el encargado de detectar el movimiento de la red. Aunque es algo que no se especifica dentro de su RFC, la forma más sencilla es hacer uso del mismo procedimiento que los terminales en la solución MIPv6, es decir mediante las facilidades que proporciona el mecanismo de descubrimiento de vecinos de IPv6 (*Neighbour Discovery*). Por ejemplo, el MR escucha los anuncios de router o *routers advertisements* (RAs) recibidos en su interfaz exterior. El MR, como todo router IPv6, también envía este tipo de mensajes pero sólo dentro de su red móvil, a través de sus interfaces internas, dado que no tendría sentido hacerlo en la red visitada.

Una vez el MR se conecta a una nueva red y recibe un RA, se autoconfigura una dirección que es topológicamente correcta dentro de la red visitada: su CoA. Inmediatamente, tras ellos, informa de este suceso a su agente local (HA), situado en su red hogar, a través de un mensaje de actualización de asociación (*Binding Update*, BU). Este paquete se envía al HA, llevando como dirección origen la CoA del MR. Para más información de este mensaje y posteriores, consulte el apéndice B.

Un BU es un tipo de mensaje definido en MIPv6 y que el grupo de trabajo de NEMO ha decidido completar y extender con nuevos tipos de información para su uso dentro de protocolo de movilidad de redes. Gracias a ello, el MR puede actuar tanto como router móvil o como nodo móvil. Este BU recoge toda la información acerca del cambio de punto de acceso que sufre el MR (información requerida por el agente local). Según el escenario que se tenga, dependiendo de si el agente local conoce a priori la dirección MNP o no, el HA necesitará que este BU le proporcione más o menos información.

Cuando el HA recibe este BU, procesa su información y crea una nueva entrada en su tabla de asociaciones, donde se guarda la asociación o *binding* entre la HoA del MR, su CoA actual y MNP(s), junto con un tiempo de vida que se asigna a la relación entre ambas. El HA asiente esta asociación al MR a través de un mensaje de asentimiento de actualización de asociación (*Binding Acknowledgement*, BA), con destino la CoA del MR.

Esto significa que todo el tráfico dirigido al MR es gestionado por el HA, que se encargará de retransmitir todo el trafico dirigido a la red móvil y de igual forma con los paquetes de la red móvil al resto de entidades. Esto se consigue a través de un túnel IPv6 sobre IPv6 bidireccional, que se crea entre los extremos del HA y el MR tras el intercambio correcto de los mensajes se señalización, BU y BA, entre el MR y el HA. Uno de los extremos del túnel se corresponderá a la dirección de la CoA del MR, y el otro extremo será la dirección del HA. Además este túnel se mantiene a pesar de que se produzca un desplazamiento de la red móvil. No es necesario destruir el túnel, sino que se reconfigura, tanto por parte del agente local como del router móvil, el extremo perteneciente al MR con la nueva CoA con que éste se configure, tras recibir el correspondiente mensaje BU del MR.

En la Figura 2.3 se muestra el escenario comentado con el correspondiente intercambio de mensajes. En el caso del tráfico generado desde un CN a un MR, se observa que el paquete llega en primer lugar a la red hogar del MR donde el HA lo encapsula , añadiéndole una cabecera IPv6 con dirección origen la suya y dirección destino la CoA del MR. A continuación el HA lo envía por el túnel hacia el MR. Una vez recibido, el MR lo desencapsula y entrega al nodo destino dentro de la red móvil.

De igual forma ocurre con el tráfico generado del MR hacia un CN, y en el escenario descrito en la Figura 2.3. Ahora el nodo perteneciente a la red móvil, envía datos hacia el nodo corresponsal. El MR se encarga de encapsular esos datos dentro un cabecera IPv6, con dirección origen la CoA del MR y con dirección destino la del HA. El paquete llega al HA a través del túnel, el cual a su vez desencapsula esa cabecera, redirigiendo el paquete al CN correspondiente.

Existen otro tipo de mensajes de señalización, *Binding Refresh*. En realidad, son mensajes de BU, pero que sirven para refrescar la entrada del MR en la tabla *Binding Cache* que posee el agente local y que permite saber que el enlace/túnel entre el MR y el HA se mantiene todavía activo, y no se encuentra caído. Por motivos de claridad, este mensaje no se muestra en la Figura 2.3, pero como ya se ha comentado es exactamente

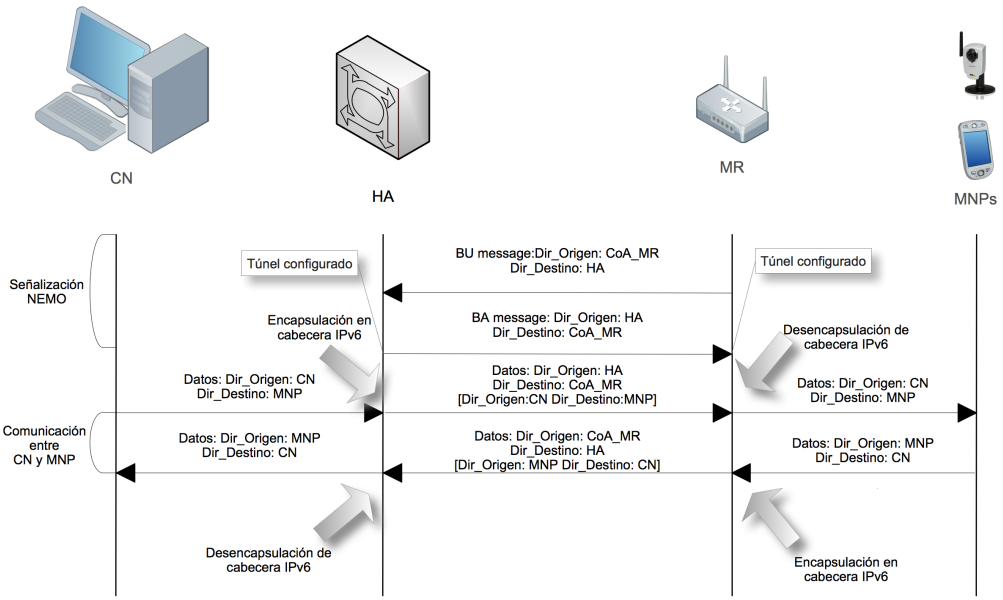

Figura 2.3: Intercambio de mensajes en NEMO.

igual que un BU.

Estos dos casos tienen lugar cuando el MR se encuentra fuera de la red hogar. Cuando regresa a la misma, ya que ha recibido el correspondiente RA procedente del HA, el proceso de señalización es el mismo, mediante el intercambio de los mensajes BU y BA. Si este intercambio es favorable, el túnel se destruye y el tráfico entre nodos del MR y diversos CN ya no es encapsulado en cabeceras IPv6. Se realizan los mecanismos estándar de encaminamientos sin que el HA ni el MR realicen mecanismos adicionales.

A modo resumen, en este capítulo se ha explicado el tema de la movilidad de redes, detallando la problemática que conlleva y las posibles soluciones que se están llevando a cabo, siendo un asunto de especial interés, debido a la relevancia e importancia que tiene e la actualidad en diversos campos, como por ejemplo el aumento de la seguridad en redes vehiculares, reduciendo el número de accidentes.

Por ese motivo, la solución propuesta por el IETF es el protocolo NEMO, cuyo funcionamiento ha sido explicado en este capítulo, al igual que la principal terminología asociada.

2.4. Redes Vehiculares

En la actualidad tiene especial relevancia el tema de las redes vehiculares, no hace falta más que comprobar la rápida proliferación de consorcios formados por fabricantes de coches, instituciones y empresas, como es el caso del consorcio de comunicaciones *Car-2-Car*, C2C-CC [car], o el de seguridad vehicular, *Vehicle Safety*, VSCCC [veh].

Una red vehicular (Figura 2.4) está formada por terminales ubicados en vehículos en movimiento, y no es sino un caso particular de red móvil. Se trata de un tipo de sistemas con numerosas aplicaciones de gran potencial económico, aún por desarrollar, como el acceso a Internet desde medios de transporte colectivos o desde redes de área personal, los servicios

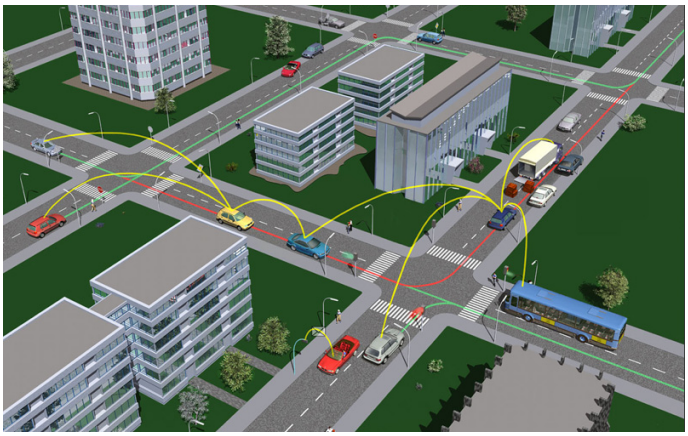

Figura 2.4: Escenario de una red vehicular.[car]

de información y asistencia en ruta al conductor, el comercio móvil, la distribución de contenidos o la conducción segura e inteligente [SBBC09].

La motivación original para el estudio de las redes vehiculares (VANET) fue aumentar la seguridad del tráfico en las carreteras, pero, hoy en día, aparecen nuevas opciones basadas en el uso de Internet para aplicaciones de entretenimiento. El protocolo NEMO BS puede aplicarse a este tipo de redes de distintas formas, obteniendo diferente impacto de acuerdo a términos económicos, funcionales y de rendimiento. Un enfoque en línea con los requerimientos de la automoción consiste en aplicar IPv6 y NEMO BS(Figura 2.5, propuesta de C2X) por encima de mecanismos *Geocast* [2] (evaluado en [BZFL08]) [OW09].

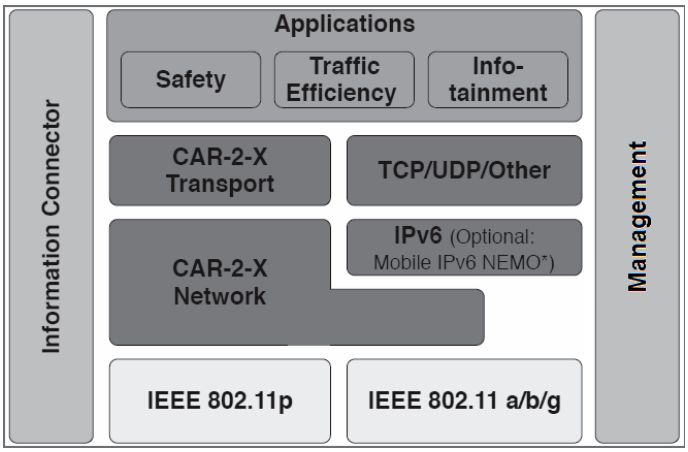

Figura 2.5: Arquitectura del protocolo CAR-2-X.

Otras líneas de investigación [BA07] proponen soluciones de integración de los protocolos MANET y NEMO: *MANET-centric* y *NEMO-centric*. Para ello se considera la red vehicular como una red móvil ad hoc convencional (VANET como MANET).

Actualmente existen diversos proyectos que investigan el tema de las redes vehiculares tales como PRE-DRIVE C2X (2008-2010), Geonet (2008 - 2010), INTERSAFE-2 (2008-2011), SIM-TD (2009 - 2011), NoW (2006-2010) etc. Estos proyectos (Figura 2.6) tratan

[2] Geocast: Mecanismo que permite la entrega de información a un grupo de destinatarios de una red identificados por su localización geográfica. Es una forma de direccionamiento multicast utilizado por algunos protocolos de enrutamiento para redes móviles ad hoc.

de dotar de capacidad de comunicación a los coches para la transmisión de mensajes, los cuales pretenden mejorar la seguridad en las carreteras.

Figura 2.6: Diferentes proyectos y consorcios internacionales relacionados con las redes vehiculares.

Todo esto da una idea de la importancia de la investigación y desarrollo en el campo de las redes vehiculares. Y además es destacable la implicación directa del protocolo NEMO BS en este tipo de redes.

Capítulo 3

Desarrollo de un prototipo de NEMO BS

3.1. Introducción

Este capítulo trata el desarrollo de una implementación básica del protocolo NEMO, a la que posteriormente se añadirán nuevas funcionalidades. En primer lugar, se presentan los distintos elementos utilizados para el desarrollo del prototipo para NEMO BS, tales como los routers utilizados y la distribución que se ha instalado en lugar de su *firmware* original. A continuación se evalúan posibles tecnologías para el desarrollo del mismo y se hará un mayor hincapié en el diseño y funcionalidad del prototipo. Finalmente se presentará la interfaz gráfica realizada para la demo del proyecto español POSEIDON [pos].

3.2. Plataforma de desarrollo

3.2.1. La distribución OpenWrt

OpenWrt (Figura *C*.1) es un *firmware* libre, basado en GNU/Linux optimizado para routers con un *hardware* determinado y reducidas capacidades. Este proyecto comenzó en 2004, basándose en un primer momento en el código fuente del router Linksys WRT54G, como mera referencia. Pero pronto se comenzó a dar soporte a modelos parecidos, incluso de otros fabricantes y de ahí a diferentes arquitecturas y *chipsets* [ope]. En el Apéndice C, concretamente en la sección C.1, se presenta más detalladamente dicha distribución. Para el desarrollo de aplicaciones destinadas a routers con este *firmware* es necesario instalar en un PC el kit de desarrollo de aplicaciones, o mejor dicho el compilador cruzado, ya que no se puede instalar en los routers un compilador debido a su escasez de memoria. Además se compila una imagen del *firmware* para incluir en ella opciones que no se encuentran en la imagen por defecto. En el Apéndice E se explica en mayor detalle los la instalación y uso de la plataforma de desarrollo de OpenWrt.

Cabe destacar que se elige la distribución *Kamikaze*, ya que ésta soporta un kernel Linux 2.6, requisito indispensable para dar soporte para túneles IPv6 sobre IPv6. La creación de estos túneles a su vez es imprescindible para poder implementar el protocolo NEMO BS, debido a que todo el tráfico dirigido a la red móvil va encapsulado a través de un túnel

con la red hogar.

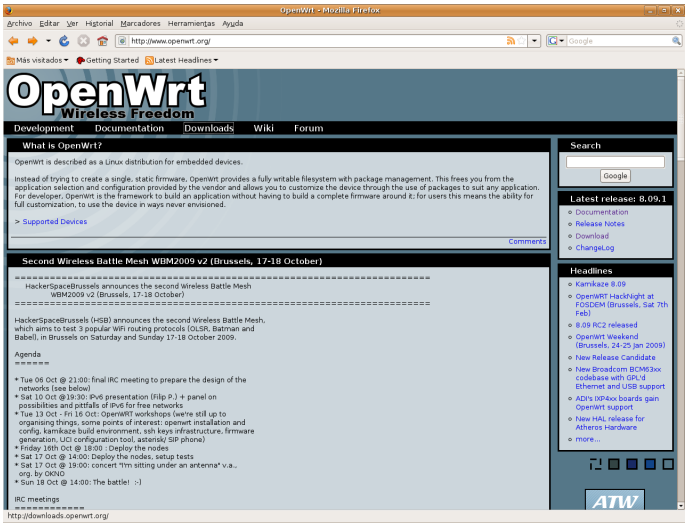

Figura 3.1: Página web de OpenWrt

3.2.2. Router Fonera

El router Fonera (Figura 3.2) surge a partir del proyecto FON. El router, modelo FON 2200, viene actualmente con el *firmware* 2.4.x OpenWrt modificado por defecto. Se comentan en mayor detalle las características de dicho router en el Apéndice C. La función que desempeña en el escenario es de router móvil.

Figura 3.2: Router FON 2200.

3.2.3. Router Linksys

El modelo WRT54G es uno de los routers inalámbricos más comercializados. Se le modifica el *firmware* original instalándose OpenWrt en su lugar. Se comentan en mayor detalle las características de dicho router y el proceso de cambio del *firmware* en el Apéndice D.

La función que desempeña en el escenario es de router de acceso, que envía *Routers Advertisements* para anunciar su prefijo de red y así el MR pueda conectarse a la red visitada.

3.3. Estructura del *software* implementado

Primeramente se va a realizar un estudio de las distintas tecnologías existentes para poder llevar a cabo la solución requerida. A continuación, se describe la solución implementada para el *software* del agente local (HA) y del router móvil (MR). Destacar que el *software* se desarrolla en lenguaje C y se hace uso de las librerías realizadas en el proyecto [AC07], que facilitan el levantamiento del túnel y la configuración de rutas basadas en *Netlink socket*, que es un método de comunicación entre el Kernel y el espacio de usuario.

3.3.1. Evaluación de las distintas tecnologías

En esta parte se evalúan distintas posibles tecnologías para llevar a cabo la implementación del prototipo [Ste90][Han04].

En primer lugar se va a definir lo qué es un *socket*. Un *socket* es el componente básico para la intercomunicación de procesos a través de la red. Los *sockets* se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de *sockets* (API, *Application Programming Interface*). La interfaz del *socket* proporciona acceso a los protocolos de transporte. Existen varios tipos de *socket* que se comentan a continuación:

- *Stream socket*: proporciona una comunicación orientada a conexión, en la que los datos se transmiten en orden y de forma secuencial, con datos no duplicados. El protocolo de transporte que usa es TCP, por lo que se establece en primer lugar una conexión entre un par de *sockets*. Mientras uno de los *sockets* atiende peticiones de conexión (servidor), el otro solicita una conexión (cliente). Una vez que los dos *sockets* estén conectados, se pueden utilizar para transmitir datos en ambas direcciones.

- *Datagram socket*: son un servicio de transporte sin conexión. Son más eficientes que TCP, pero en su utilización no está garantizada la fiabilidad. Los datos se envían y reciben en paquetes, cuya entrega no está garantizada. El protocolo de transporte que utiliza es UDP. Cada vez que se envíen datagramas es necesario enviar el descriptor del *socket* local y la dirección del *socket* que debe recibir el datagrama.

- *Raw socket*: son *sockets* que dan acceso directo a la capa de *software* de red subyacente o a protocolos de más bajo nivel. Se utilizan sobre todo para la depuración del código de los protocolos.

Por tanto, se decide que los *raw sockets* son los que se van a utilizar en este proyecto ya que se desea tener acceso a nivel IP. Además el hecho de utilizar este tipo de *sockets* da la libertad de crear la estructura del paquete IP tal y como será retransmitido por la red, sin que el kernel añada o modifique alguna cabecera. Comentar que UNIX provee de estructuras estándar en lenguaje C para las cabeceras de distintos protocolos TCP/IP, y así se puede acceder a los campos de cada cabecera de forma más cómoda.

3.3.2. Agente Local

En líneas generales, la estructura de la implementación para el funcionamiento de un equipo como agente local se presenta en el diagrama de flujo de la Figura 3.3. Como se puede ver, a grandes rasgos la estructura es muy sencilla: se comienza creando la memoria de asociaciones o ***B**inding **C**ache* y extrayendo la información necesaria de un fichero de configuración inicial. Esta información corresponde a los routers móviles asociados a ese agente local y sus correspondientes MNPs.

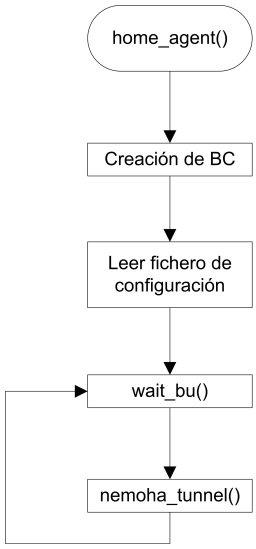

Figura 3.3: Diagrama de flujo de la función *home_ agent()*

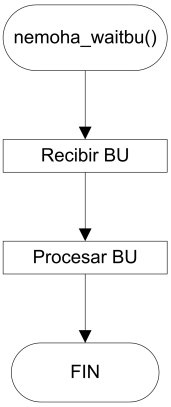

Figura 3.4: Diagrama de flujo de la función *wait_ bu()*

A continuación la función wait_bu() (Figura 3.4) se encarga de preparar la recepción de mensajes de tipo *Binding Update*. Dependiendo de si el BU recibido pertenece a un router móvil que se encuentra ya registrado o no, se procederá de una forma u otra. Pero esto se explica con mayor detalle en la Figura 3.5.

Y en el momento en que se recibe uno de estos mensajes, la función tunnel() (Figura 3.5) hace todo lo necesario para la configuración del túnel hacia el MR. En primer lugar se comprueba si el router móvil se encontraba ya o no registrado en la *Binding Cache*. En caso se crea una nueva entrada en la tabla de asociaciones. Se procede a enviar un mensaje de asentimiento, *Binding ACK*, se configuran el túnel y las correspondientes rutas.

En el caso de que el router móvil ya esté registrado, se comprueba si la dirección (CoA) del paquete es la misma con la que se registró el MR en la *Binding Cache*, si es así se actualiza el tiempo de vida o *lifetime* y número de secuencia de esa entrada en la tabla. En caso contrario, esto significa que el router móvil ha cambiado de punto de acceso, por lo que se modifica la entrada del MR, tanto la dirección o CoA, como el tiempo de vida en la tabla y número de secuencia. También se reconfigura el túnel, para permitir que el MR siga siendo alcanzable.

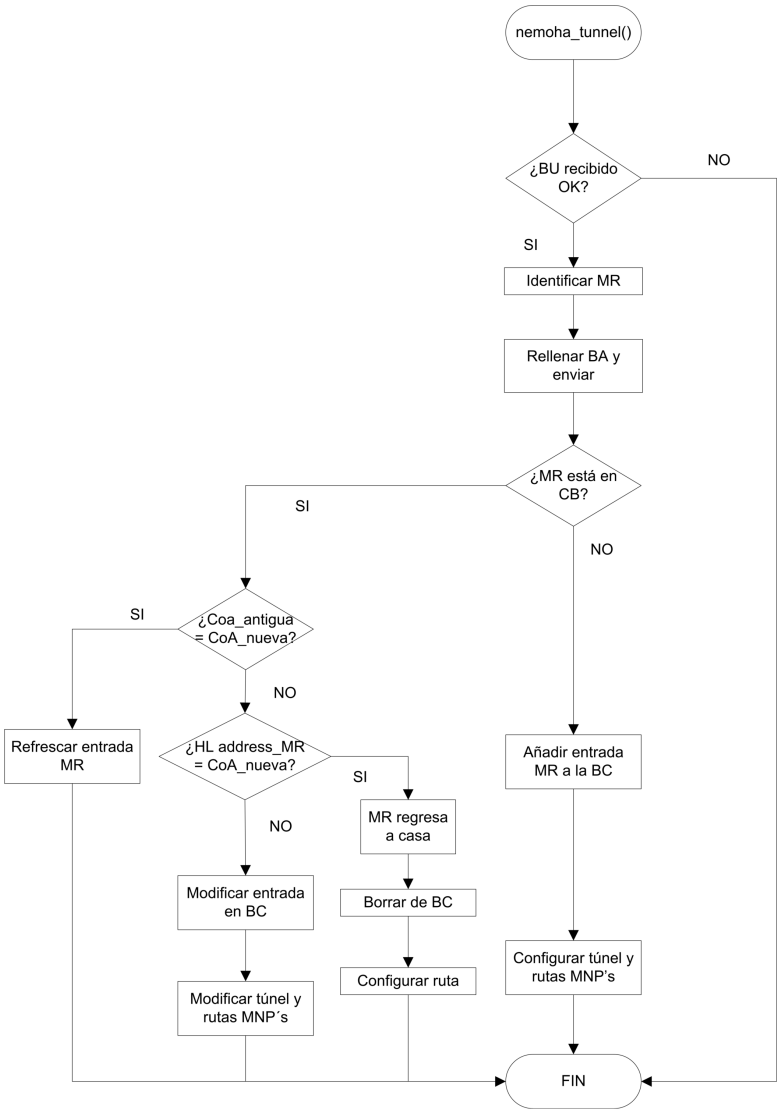

Figura 3.5: Diagrama de flujo de la función *nemoha_ tunnel()*

Tras finalizar la configuración, todo el tráfico dirigido al MR o a alguno de los nodos pertenecientes a la red móvil serán encaminados a través del túnel, que tiene como extremos la dirección del HA y la CoA del MR. Por su parte, el agente local vuelve a prepararse para la recepción de nuevos mensajes *Binding Update*.

En el caso de que el tiempo de vida de alguna de las entradas de la *Binding Cache* llegue a cero, se procederá a borrar esa entrada de la tabla y a la consiguiente eliminación de rutas y túnel asociados a ese router móvil.

En el momento en que el *software* recibe una señal para parar la ejecución del programa, la función stop() (Figura 3.6) se encarga de borrar las rutas y el túnel dedicados a los diversos MRs que haya asociados, liberando los recursos utilizados, y finaliza la aplicación.

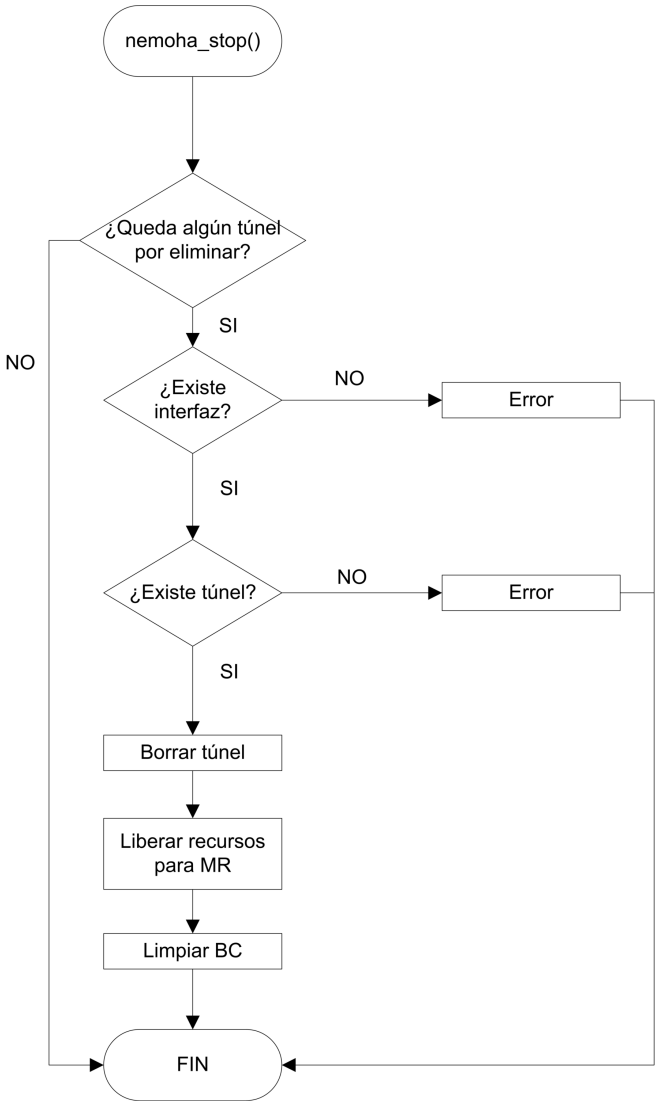

Figura 3.6: Diagrama de flujo de la función *nemoha_ stop()*

3.3.3. Router Móvil

En líneas generales, la estructura de la implementación para el funcionamiento de un equipo como router móvil se presenta en el diagrama de flujo de la Figura 3.7. Como se puede ver, a grandes rasgos la estructura es muy sencilla: se comienza borrando la memoria de asociaciones o ***Binding Cache*** y extrayendo la información necesaria de un fichero de configuración inicial, referente a sus interfaces *ingress* y *egress*, sus prefijos (MNPs), y su HoA.

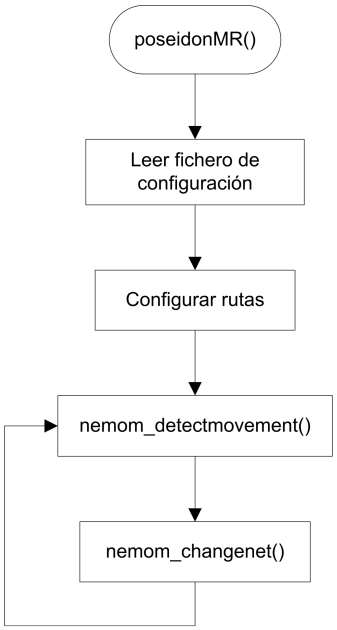

Figura 3.7: Diagrama de flujo de la función *poseidon_ MR()*

Tras esto, se detecta el movimiento a partir de la recepción de anuncios de router o *Router Advertisements* (Figura 3.8). El router móvil ignora sus propios RAs; cuando le llega uno recibido por su interfaz *egress* comprueba si el prefijo del mismo es igual al de su dirección CoA. Si es así, simplemente el MR envía un mensaje *Binding Refresh* (ver Apéndice B, sección B.4) al agente local, para que éste refresque su entrada en la *Binding Cache*. El MR vuelve al estado de capturar RAs.

En el caso de que el prefijo del *Router advertisement* sea diferente al de su CoA, esto quiere decir que se ha producido un cambio de red, por lo que el MR se encuentra asociado a router de acceso distinto. Primero comprueba si se trata de la red hogar, en este caso borra el túnel y distintas rutas asociadas. En caso de que se trate de una red visitada, modifica el túnel y las rutas, enviando un mensaje *Binding Update* al agente local para notificarle el cambio de red y que éste pueda reconfigurar su extremo del túnel y rutas, respectivamente. En este momento, el MR espera al envío del *Binding ACK* por parte del HA. El túnel no se crea hasta que el MR recibe este mensaje. El reenvío de mensajes BU se basa en el mecanismo del algoritmo *exponential backoff*, el cual aumenta el tiempo de retransmisión del mensaje de forma exponencial binaria. El router móvil envía un máximo de 6 mensajes BU, que es el número de mensajes que da tiempo a enviar en 32 segundos (MAX_BINDACK_TIMEOUT) [JPA04], en el caso de no recibir respuesta por parte del agente local termina la ejecución de la aplicación.

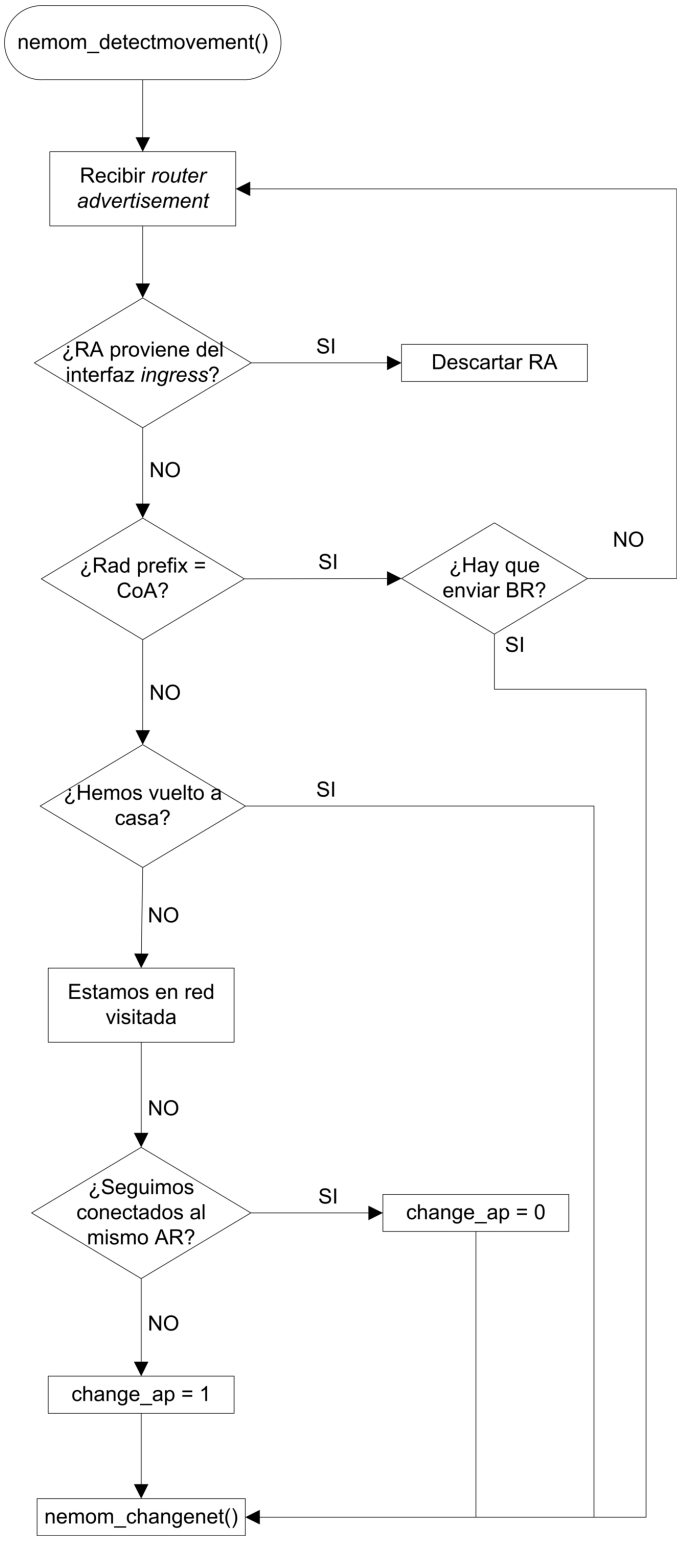

Figura 3.8: Diagrama de flujo de la función *detect_ movement()*

En el momento en que el *software* recibe una señal para parar la ejecución del programa, la función stop() (Figura 3.9) se encarga de borrar las rutas y el túnel dedicados al HA, liberando los recursos utilizados, y finaliza la aplicación.

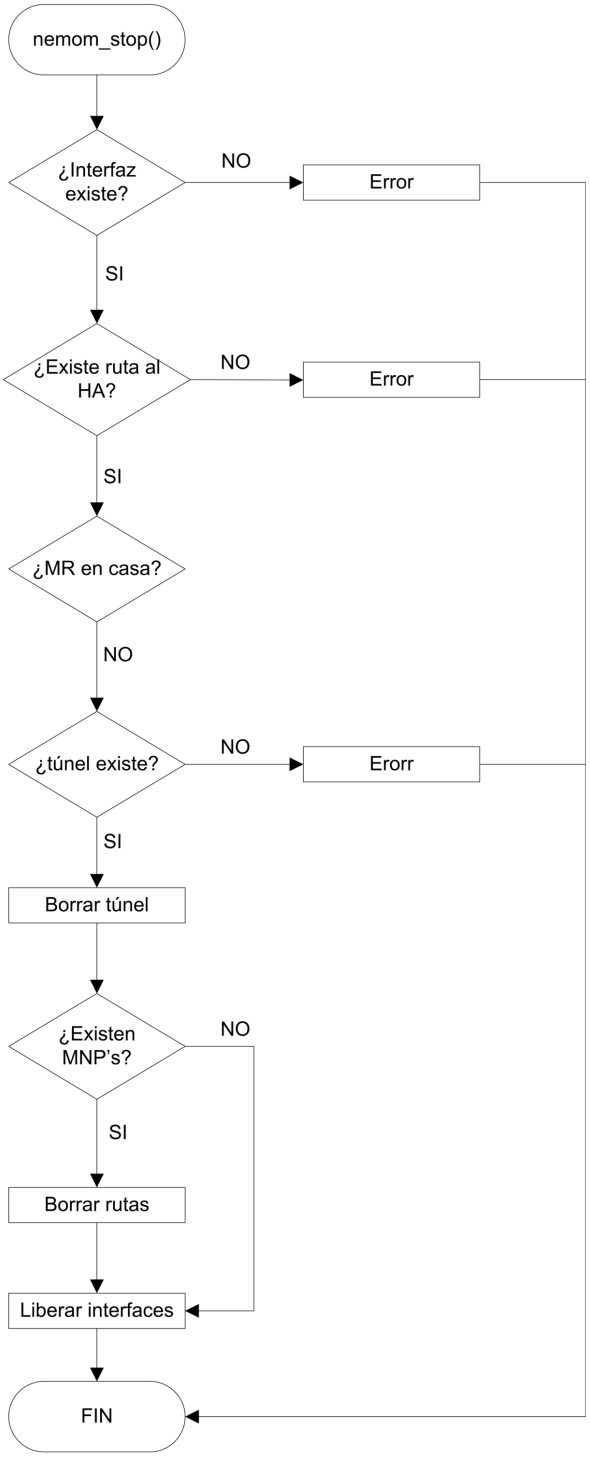

Figura 3.9: Diagrama de flujo de la función *nemom_ stop()*

3.4. Consideraciones adicionales

- Para eliminar el mensaje de error (*ICMPv6 Parameter Problem*) que se obtiene al recibir cualquiera de los mensajes de señalización, se debe incluir en el kernel el módulo de soporte de movilidad a nivel IPv6. Simplemente se han de seleccionar las opciones relativas a la movilidad IPv6 (*IPv6: Mobility Support (EXPERIMENTAL)*

(m) y *MIPv6: Debug Messages (m)*). Este módulo se ha podido incluir en el caso del agente local, sin embargo para el router Fonera dicho módulo de movilidad a nivel IPv6 no se encuentra implementado para la distribución *Kamikaze*, sin embargo sí está soportado en la nueva línea de desarrollo de OpenWrt, *Backfire*.

- Una de las pruebas que interesa realizar consiste en enviar tráfico a distintas tasas y con diferentes tamaños de paquete, para comprobar la influencia de la señalización y en que medida la información de control añadida disminuye el rendimiento. El desarrollo de estas pruebas se describe en la sección 5.6 del siguiente capítulo. Por esta razón se desactiva la opción *encapsulation limit* (número máximo de túneles anidados) a la hora de crear el túnel, estableciéndola a *none*. De otro modo, esta opción añadiría 8 bytes a la cabecera de IPv6 que encapsula los datagramas a través del túnel, reduciendo el tamaño útil de los paquetes.

3.5. Conclusiones

Para llevar a cabo la implementación se realizó una exhaustiva evaluación de las distintas tecnologías de programación de *sockets*, así como de las especificaciones del protocolo. También se encontraron diversos problemas al utilizar *raw sockets* y tener que acceder a datos de la cabecera, por ello se utilizaron las opciones de movilidad para poder enviar la CoA (*Alternate Care-of Address*), explicadas en el Apéndice B sección B.5. Al igual que la opción *Home Address Destination Option*, Apéndice B sección B.7, que era necesario incluir para que el nodo móvil mientras esté fuera de casa informe al agente local de su dirección en la red hogar. Estas dos opciones permiten al HA acceder a toda la información que necesita para gestionar la movilidad de los MRs que pertenezcan a su red.

Tras haber realizado la implementación del protocolo, es necesario realizar una batería de pruebas, tanto cualitativas como cuantitativas, para evaluar el prototipo y la incidencia de este protocolo en el rendimiento.

Capítulo 4

Desarrollo de un demostrador

En este capítulo se va a presentar el demostrador realizado para el prototipo, el cual ha sido utilizado en el proyecto español POSEIDON ***P**rovisión **Ó**ptima de **SE**rvIcios a re**D**es vehiculares en m**O**vimie**N**to* (TSI2006-12507-C03). Se van a comentar en detalle las herramientas de visualización, la parte del desarrollo de una red vehicular, la elección de distintos dispositivos que forman parte de la misma, tales como la cámara IP o los coches radio control. También se describe la alimentación autónoma de los dispositivos, ya que era requisito indispensable para poder simular un escenario de comunicaciones móviles.

4.1. Introducción a POSEIDON

El proyecto POSEIDON[1] es un proyecto financiado por el Ministerio de Educación y Ciencia dentro del programa de Tecnologías de Servicios de la Sociedad de la Información (TSI2006-12507-C03), realizado por la Universidad Carlos III de Madrid, la Universidad de Vigo y la Universidad Politécnica de Cataluña.

La finalidad de POSEIDON es la creación de soluciones de ingeniería para la provisión de servicios de calidad sobre redes heterogéneas destinados a usuarios conectados a redes móviles, muy particularmente a redes vehiculares (Figura 4.1).

Además del continuo aumento en escala, Internet debe asumir la doble presión de proporcionar servicios con un grado de movilidad cada vez mayor y de acoger en su estructura tecnologías cada vez más diversas tanto en el tramo de acceso (en especial, con distintos tipos de accesos inalámbricos) como en el propio núcleo de transporte de los datos (donde, no importa la longitud de la troncal, se impone la tecnología óptica). El objetivo de este proyecto es investigar, analizar y evaluar soluciones a los desafíos técnicos que provienen simultáneamente de la demanda de movilidad en las redes de acceso y de la existencia de una heterogeneidad creciente así en las subredes de acceso como en la infraestructura de Internet. A tal fin, el proyecto se plantea el desarrollo de soluciones de ingeniería para la provisión óptima de servicios telemáticos a redes vehiculares en movimiento. Este tipo de plataformas móviles de acceso a Internet encierran un gran potencial de desarrollo de nuevos servicios, al tiempo que constituyen un avanzado campo de juego idóneo para hacer avanzar la tecnología básica que interviene en la mejora de funcionalidad del soporte a usuarios en movimiento, o en el refinamiento de un conjunto de

[1] http://enjambre.it.uc3m.es/ poseidon/

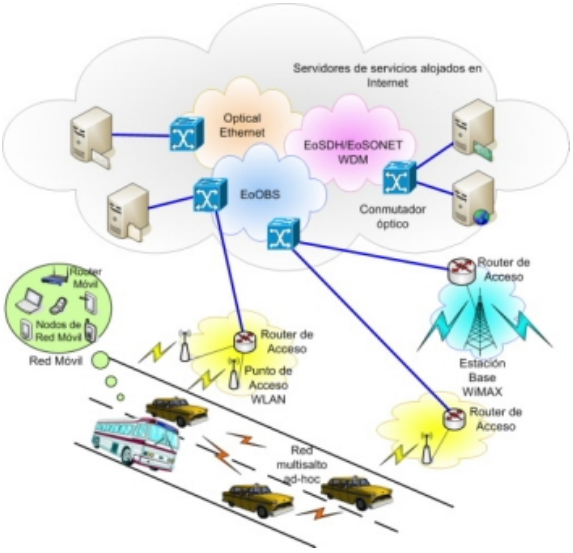

Figura 4.1: Escenario propuesto por POSEIDON.

algoritmos y protocolos de soporte de optimización de recursos (rutas y ancho de banda) sobre conexiones extremo a extremo heterogéneas.

De manera más concreta, el proyecto se estructura en tres áreas de trabajo fundamentales:

- Provisión de servicios en la parte de red móvil, centrada en el desarrollo de tecnología de soporte de la movilidad (optimización de rutas), provisión de calidad de servicio en entornos inalámbricos multisalto o de un solo salto, gestión de los traspasos de usuarios y manejo de múltiples interfaces de acceso a red. Se considerará explícitamente la provisión de servicios de comunicaciones directos entre usuarios móviles.
- Provisión de servicios en la parte de red fija, dedicada fundamentalmente a resolver los problemas del diseño de un plano de control inteligente (elección de rutas, asignación y reserva de recursos, señalización, planificación, etc.) que permita una utilización óptima del ancho de banda extremo a extremo.
- Provisión de tecnologías de soporte de la heterogeneidad en las partes fija y móvil, e interacción red fija-red móvil. En esta área se abordarán los aspectos de análisis y desarrollo de nuevos mecanismos de transporte, de diferenciación de servicios, de planificación, de balanceo de carga y de control de la congestión que tomen en cuenta la diferente naturaleza de las distintas partes de la red global, de suerte que, pese a la mayor complejidad del sistema, sea posible explotar de forma óptima los recursos del nivel de red y las capacidades provistas de forma específica por los canales inalámbricos y los canales de transmisión/conmutación ópticos.

4.2. Plataforma de pruebas

En este apartado se describe la plataforma de pruebas desarrollada que simula una red vehicular. El despliegue de esta plataforma era necesario de acuerdo a los requerimientos

exigidos por el proyecto POSEIDON, permitiendo además tener una aproximación más real para evaluar el prototipo realizado, ya que es posible realizar el traspaso a nivel 2, es decir, por nivel de señal, no como han tenido lugar las pruebas, forzando el cambio de punto de acceso.

El elemento principal de esta plataforma de pruebas es la red móvil (Figura 4.2), que se compone de los siguientes elementos:

- Un coche radio control, que emula a un vehículo.
- Un router Fonera 2200, que actúa como router móvil.
- Una cámara IP, que actúa como un nodo móvil. La sección 4.4 detalla las características y uso de este dispositivo, y la elección del modelo de cámara utilizado.

Figura 4.2: Coche RC con cámara IP.

El hecho de tener una red móvil hace indispensable alimentar los elementos de forma autónoma. Este punto se describe con mayor detalle en la sección 4.5.

4.3. Interfaz gráfica

Para tener una visión global del funcionamiento de la aplicación, se realiza una versión gráfica que permita rastrear el movimiento de la red móvil.

Para ello se utiliza el programa Kommander[2]. Esta aplicación se compone de dos partes, la primera es un editor (Figura 4.3) donde se puede visualizar el cuadro de diálogos, formado por los distintos elementos que forman la versión gráfica. También permite editar el *script* que simulará el movimiento del vehículo. Para ello se lee la *Binding Cache* del agente local y se comprueba a qué punto de acceso se encuentra asociado el vehículo (MR), a partir de su dirección CoA, y con ello simular el movimiento o no.

[2] http://kommander.kdewebdev.org/

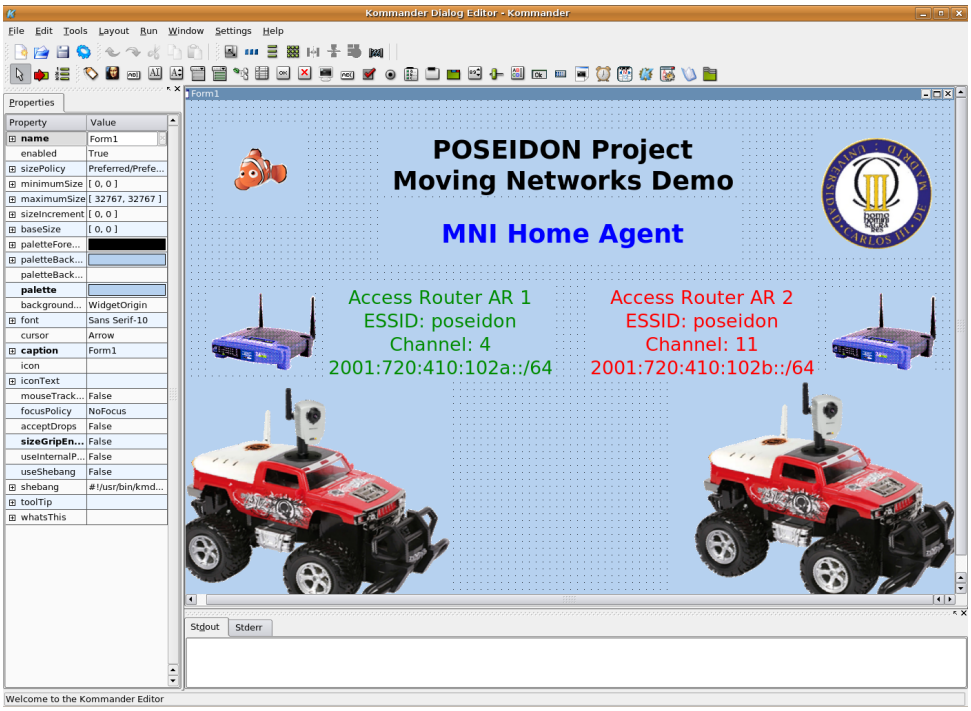

Figura 4.3: Vista del editor de Kommander

La segunda parte de esta aplicación es un ejecutor que procesa el fichero XML (*extensible Markup Language*) generado, mostrando la interfaz creada. Las Figuras 4.4, 4.5 y 4.6 muestran la ejecución de la versión gráfica cuando se produce un cambio de punto de acceso por parte del vehículo, simulando el movimiento del mismo.

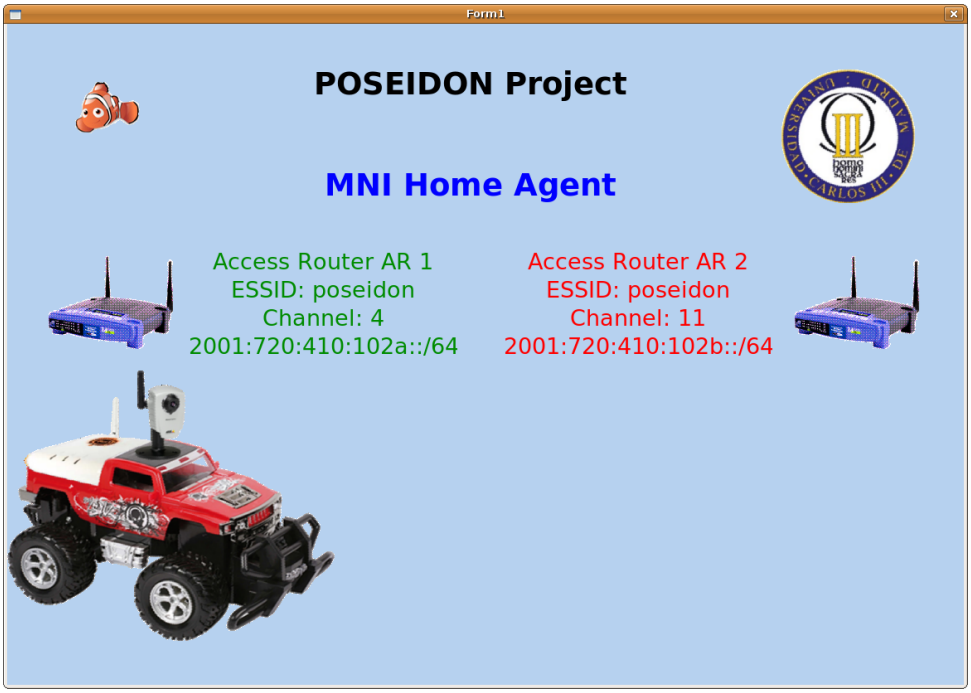

Figura 4.4: Vista de la versión gráfica para un traspaso entre puntos de acceso (I)

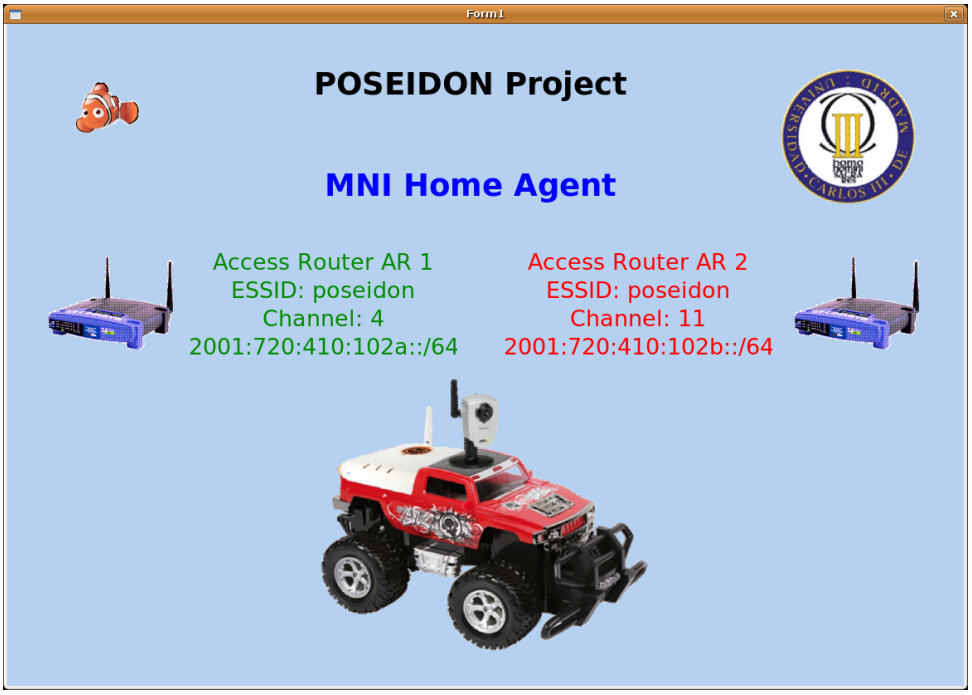

Figura 4.5: Vista de la versión gráfica para un traspaso entre puntos de acceso (II)

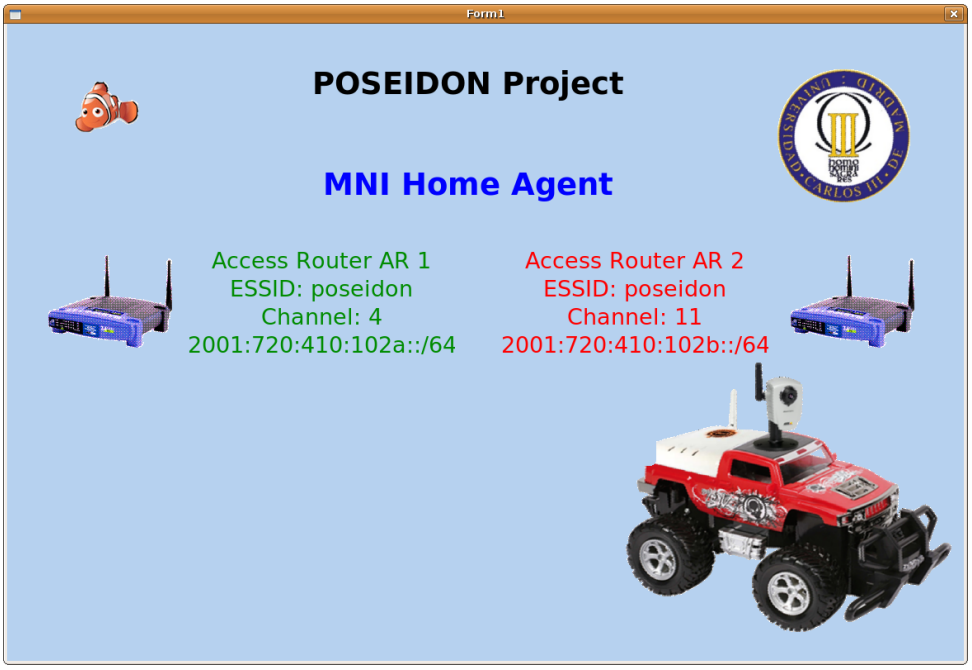

Figura 4.6: Vista de la versión gráfica para un traspaso entre puntos de acceso (III)

4.4. *Cámara IP*

En este proyecto se ha utilizado la cámara IP Axis 207W (Figura 4.7) para enviar vídeo y audio en tiempo real desde la red móvil. Una cámara IP es un dispositivo que permite visionar en directo, no sólo imágenes sino también vídeo, procedente de cualquier parte del mundo. Entre sus distintas aplicaciones se encuentra la vigilancia de hogares y empresas, con la posibilidad de almacenar el vídeo en una localización remota a la que sólo la persona autorizada tenga acceso. La función que desempeña la cámara IP en este proyecto es la de nodo móvil encargado de enviar contenido multimedia a un nodo corresponsal (CN).

Figura 4.7: Cámara IP AXIS 207W.

Tras realizar un estudio de la situación actual de este tipo de cámaras en el mercado, se confió en el fabricante Axis, dado el gran número de modelos distintos que ofrece, en relación a otros más conocidos, como Canon o Sony. Entre los distintos modelos ofrecidos, se escogió el 207W por distintas características: reducido tamaño, presencia de un interfaz inalámbrica y más importante aún, soporte del protocolo IPv6. En la Figura 4.1 se muestra una comparativa de los distintos modelos de cámaras IP ofrecidos por Axis.

	AXIS 207[a] AXIS 207W[b]	AXIS 207MW	AXIS M1011[a] AXIS M1011-W[b]	AXIS M1031-W	AXIS 210[a] AXIS 210A[b]	AXIS P1311	AXIS 211[a] AXIS 211A[b]	AXIS 211W	AXIS 211M	AXIS 221	AXIS P1343	AXIS 223M	AXIS Q1755
Image sensor	1/4" progressive scan CMOS	1/3" progressive scan CMOS	1/4" progressive scan CMOS	1/4" progressive scan CMOS	1/4" progressive scan CCD	1/4" progressive scan CMOS	1/4" progressive scan CCD	1/4" progressive scan CMOS	1/3" progressive scan CMOS	1/3" progressive scan CCD	1/4" progressive scan CMOS	1/2.7" progressive scan CCD	1/3" progressive scan CMOS
Lens	4 mm/F2.0 fixed iris	3.6 mm/F1.8 fixed iris	4.4 mm/F2.0 fixed iris	4.4 mm/F2.0 fixed iris	4 mm/F1.2 fixed iris, CS mount	4 mm/F1.2 fixed iris, CS mount	Varifocal 3 – 8 mm/F1.0 DC-iris, CS mount	Varifocal 3 – 8 mm/F1.0 DC-iris, CS mount	Varifocal 3 – 8 mm/F1.0 DC-iris, CS mount	Varifocal 3 – 8 mm/F1.0 DC-iris, CS mount	Varifocal 3 – 8 mm/F1.4 DC-iris, CS mount Remote back focus	Varifocal 4 – 8 mm/F1.4 DC-iris, C mount[1]	5.1 – 51 mm/F1.8 auto iris and autofocus 10x optical zoom 12x digital zoom
Horizontal angle of view	55°	74°	47°	47°	48°	42°	27° – 67°	27° – 67°	37° – 93°	35° – 93°	25° – 59°	38° – 72°	5.4° – 50°
Day and night										Automatic	Automatic	Automatic	Automatic
Min illumination/ light sensitivity (lux)	1 – 10,000	2 – 10,000	1 – 10,000	1 – 10,000 0 with LED on	1 – 10,000	0.6 – 30,000	0.75	0.75	1	0.65 (color) 0.08 (B/W)	0.3 (color) 0.05 (B/W)	1.5 (color) 0.2 (B/W)	2 (color) 0.2 (B/W)
Video compression	MPEG-4 Motion JPEG	MPEG-4 Motion JPEG	H.264 Motion JPEG MPEG-4	H.264 Motion JPEG MPEG-4	MPEG-4 Motion JPEG	H.264 Motion JPEG MPEG-4	MPEG-4 Motion JPEG	MPEG-4 Motion JPEG	MPEG-4 Motion JPEG	MPEG-4 Motion JPEG	H.264 Motion JPEG	MPEG-4 Motion JPEG	H.264 Motion JPEG
Max video resolution (pixels)	640 x 480	1280 x 1024	640 x 480	640 x 480	640 x 480	640 x 480	640 x 480	640 x 480	1280 x 1024	640 x 480	800 x 600	1600 x 900 1600 x 1200	HDTV 1080i 1920 x 1080 HDTV 720p 1280 x 720
Frames per second	30 (640x480)	12 (1280x1024)	30 (640x480)	30 (640x480)	30 (640x480)	30 (640x480)	30 (640x480)	30 (640x480)	12 (1280x1024) 30 (800x600)	45 (640x480) 60 (480x360)	30 (800x600)	12 (1600x900) 9 (1600x1200)	30/25
Audio support	One-way, built-in mic	One-way, built-in mic		Two-way, built-in mic and speaker	Two-way, built-in mic[b]	Two-way, built-in mic	Two-way, built-in mic[b]	Two-way, built-in mic	Two-way, built-in mic		Two-way	Two-way	Two-way, built-in mic
Alarm in-/outputs	1/1	1/1			1/1	1/1	1/1	1/1	1/1	2/1	1/1	2/1	2 configurable inputs/outputs
Intelligent video	Motion detection Audio detection	Motion detection Audio detection	Motion detection Tampering alarm	Motion detection Audio detection Tampering alarm	Motion detection Audio detection[b] Tampering alarm[b]	Motion detection Audio detection Tampering alarm	Motion detection Audio detection[b] Tampering alarm[b]	Motion detection Audio detection	Motion detection Audio detection	Motion detection Tampering alarm	Motion detection Audio detection Tampering alarm	Motion detection Audio detection Tampering alarm	Motion detection Audio detection Tampering alarm Gatekeeper
Security	Multi-level passwords IP filtering[b] HTTPS encryption[b] IEEE 802.1X[b]	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X	Multi-level passwords IP filtering HTTPS encryption	Multi-level passwords IP filtering HTTPS encryption	Multi-level passwords IP filtering HTTPS encryption[b] IEEE 802.1X [b]	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X	Multi-level passwords IP filtering HTTPS encryption[b] IEEE 802.1X [b]	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X	Multi-level passwords IP filtering HTTPS encryption IEEE 802.1X
Network	IPv4/v6[b], QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS	IPv4/v6, QoS
Power over Ethernet (IEEE 802.3af)	PoE splitter available	PoE splitter available	PoE splitter available	PoE splitter available	AXIS 210: PoE splitter AXIS 210A: Class 2	Class 1	Class 2	Class 2	Class 2	Class 2	Class 2	Class 2	Class 3
Serial connectors										RS-232, RS-485		RS-232, RS-485	RS-422/RS-485
Outdoor use						Requires housing	Requires housing	Requires housing	Requires housing	Requires housing	Requires housing	Requires housing	Requires housing
Other	Wireless[b]	Wireless	Wireless[b]	Built-in PIR sensor Illumination LED Wireless	Changeable lens	Changeable lens SD/SDHC memory card slot	Changeable lens	Changeable lens Wireless	Changeable lens	Changeable lens	Changeable lens SD/SDHC memory card slot	Changeable lens	SD/SDHC memory card slot Analog video out Pan/tilt head support

Notations (a) and (b) refer to the corresponding product models for the column

1) Camera can be used with C/CS lens

Tabla 4.1: Tabla Comparativa Cámaras IP.

4.5. *Hardware* para alimentación autónoma

En esta sección se van a describir los mecanismos utilizados para alimentar de forma autónoma a los distintos dispositivos que forman la red móvil.

4.6. *Hardware* para alimentación autónoma del router Fonera

Debido a que este proyecto está totalmente relacionado con la movilidad, es indispensable conseguir que los distintos dispositivos móviles (MR y MNN) en efecto lo sean.

Una primera solución fue utilizar un portapilas para cuatro pilas de tipo AA, ya que cada una de ellas proporciona una tensión de 1.2 V, dando lugar a una tensión total del 4.8 V, que resulta suficiente para alimentar el router Fonera. Estas pilas se soldaron a un conector macho de alimentación, quedando finalmente como se muestra en la Figura 4.8.

Figura 4.8: Portapilas para la alimentación de la Fonera.

Como mejora, se adquirió una batería de 6V y 4A para proveer al router Fonera de una mayor autonomía (Figura 4.9). La limitación de la tensión de alimentación de dicho router es de 7V por lo que no ha sido necesario emplear ningún circuito adicional para regular esa tensión.

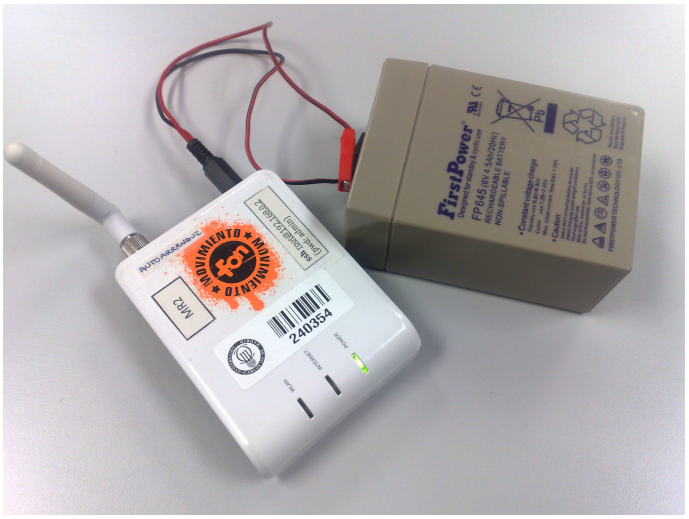

Figura 4.9: Batería para la alimentación de la Fonera.

4.7. *Hardware* para alimentación autónoma de la cámara IP

Para otros dispositivos, por ejemplo la cámara IP Axis 207W, que actúa como nodo móvil y servidor de vídeo, la tensión de alimentación está entre 4.9V y 5.1V, por lo que se hace imprescindible utilizar un circuito que limite la tensión suministrada por la batería. Para ello se emplea un dispositivo regulador de tensión de salida variable de la serie L7800, escogiéndose el L7804CV ya que la tensión de salida que proporciona es de 5V. Posee 3 terminales: entrada, salida y masa, como se puede comprobar en la Figura 4.10.

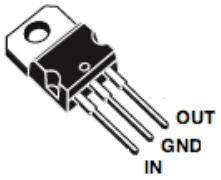

Figura 4.10: Regulador de tensión L7804CV.

El circuito de la Figura 4.11 es el que proporciona el fabricante para obtener un valor de 5 V de continua a la salida del regulador. El circuito con todos los componentes soldados se muestra en la Figura 4.12.

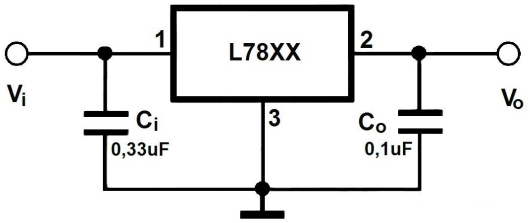

Figura 4.11: Circuito regulador de tensión para alimentar la Fonera.

De esta manera se consigue no alimentar los dispositivos con una tensión superior a la requerida, ya que pequeños valores de tensión de entrada por encima del especificado en

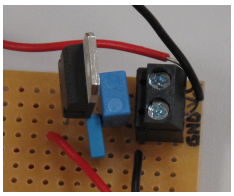

Figura 4.12: Circuito soldado regulador de tensión para alimentar la Fonera.

su hoja de catálogo pueden provocar la rotura de dichos equipos.

Este circuito regulador también se utilizó para alimentar de forma autónoma la cámara IP Axis 207W 4.7, debiendo prestar especial atención al valor de alimentación (estando entre 4.9 y 5.1 voltios). Esta forma de alimentación se realizaba a través del *jack*, como el que viene en la fuente. En la misma línea de alimentación se pueden encontrar en el mercado *battery packs* como, por ejemplo, el que se puede comprar en la página web `http://www.franz-video.de`.

En el caso de la cámara IP, otra solución para la alimentación autónoma sin tener que recurrir al conector de electricidad es utilizar un *POE - Power Over Ethernet - splitter*. Esta tecnología, cuyo estándar es el IEEE 802.3af, consiste en alimentar el dispositivo a través de la interfaz Ethernet, una infraestructura LAN estándar. Permite que la alimentación eléctrica se suministre al dispositivo de red como, por ejemplo, un teléfono IP o una cámara de red, usando el mismo cable que se utiliza para una conexión de red. Elimina la necesidad de utilizar tomas de corriente en las ubicaciones de la cámara y permite una aplicación más sencilla de los sistemas de alimentación ininterrumpida (SAL) para garantizar su funcionamiento las 24 horas del día, 7 días a la semana. Este POE se puede conseguir por ejemplo en la siguiente URL: `http://www.axis.com/products/pol/poe/index.htm`. El modelo compatible con la cámara utilizada sería el splitter 5008-5001 de 5V.

A pesar de que esta opción es más cómoda, ha sido descartada, debido a que el router Fonera, que va a funcionar como router móvil, no soporta la tecnología POE de forma activa. Es decir, que puede ser alimentado mediante un POE pero el propio router no puede alimentar la cámara IP. En proyectos posteriores en los que se usa otro tipo de routers, como es el *Asus WL-500G Premium* que posee una conexión USB, sería posible alimentar de forma autónoma la cámara IP, como se muestra en: `http://www.h-i-r.net/2010/02/guest-post-fonera-power-over-ethernet.html`

Capítulo 5

Evaluación del prototipo

5.1. Introducción

En el siguiente capítulo se presentan los resultados de las pruebas realizadas para la evaluación el prototipo.

En primer lugar se realizan unas pruebas que permitan demostrar el correcto funcionamiento del protocolo. Posteriormente, se llevan a cabo otra serie de pruebas, tanto cualitativas como cuantitativas, que muestren el rendimiento de la aplicación.

En primer lugar se van a describir los distintos escenarios de prueba establecidos, que permitirán comprobar el adecuado funcionamiento del *software* desarrollado para el agente local y el router móvil, de acuerdo a las especificaciones del protocolo NEMO BS definidas en la RFC 4886 [Ern07]. Se comprobará de igual forma que la señalización es la requerida.

A continuación en el segundo conjunto de pruebas se evaluará el rendimiento del *software*, mediante pruebas que medirán el tiempo de traspaso o *handover* y de eficiencia o *throughput* para distintos tipos de paquetes.

Por último, se aumentará la configuración de dicho escenario añadiendo un coche radio control, emulando una red vehicular, y una cámara IP, que actuará como nodo móvil conectado al router móvil. A su vez se realizarán pruebas cualitativas que permitan determinar la calidad del vídeo que está siendo retransmitido a la hora de producirse un traspaso entre distintos puntos de acceso al desplazarse el nodo móvil.

Por otra parte se realizará un estudio de la autonomía del router Fonera 2200, proporcionada mediante pilas AA y una batería.

5.2. Escenarios de prueba

Antes de pasar a comentar con mayor detalle las pruebas realizadas, se van a describir los distintos escenarios empleados para la mismas.

En primer lugar, como se pretende validar el prototipo, se va a disponer de un escenario menos complejo (Figura 5.1). Este escenario consta de un router de acceso, un agente local y un router móvil. Las direcciones de cada uno de los elementos se muestran en dicha Figura.

Con este escenario se pretende comprobar que los equipos se configuran correctamente y que la señalización al asociarse a una red visitada por parte del MR cumple los requisitos del protocolo.

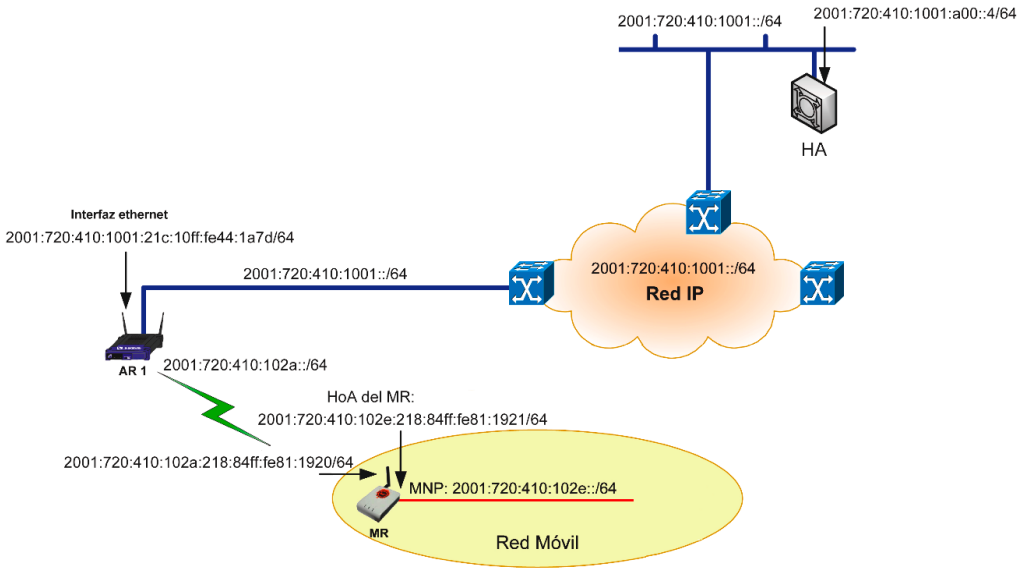

Figura 5.1: Escenario de prueba 1.

Posteriormente se eleva la complejidad del escenario, añadiendo otro router de acceso y un nodo corresponsal (Figura 5.2). Con ello se pretende comprobar el comportamiento adecuado del prototipo al producirse un cambio en el punto de acceso o traspaso en la red visitada por parte del router móvil. También se puede comprobar que el tráfico que se intercambia entre el router móvil y el nodo corresponsal se realiza a través del túnel IPv6. En paralelo, se probará también que el *software* implementado para el agente local soporta la gestión de varios routers móviles, cada uno de ellos, a su vez, gestionando el soporte de distintos nodos móviles.

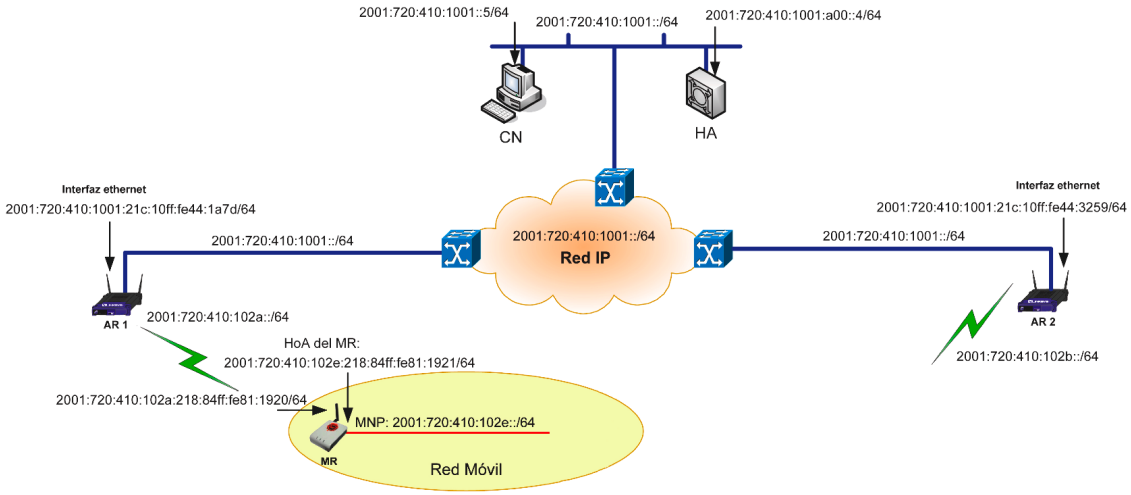

Figura 5.2: Escenario de prueba 2.

Por último, se unirán todos estos elementos junto con una cámara IP y un coche radio control, para generar un escenario completo de pruebas como se muestra en la Figura 5.3.

Este escenario tratará de simular el movimiento real de una red vehicular. Se realizarán retransmisiones de vídeo, permitiendo así realizar pruebas cualitativas de la calidad del mismo.

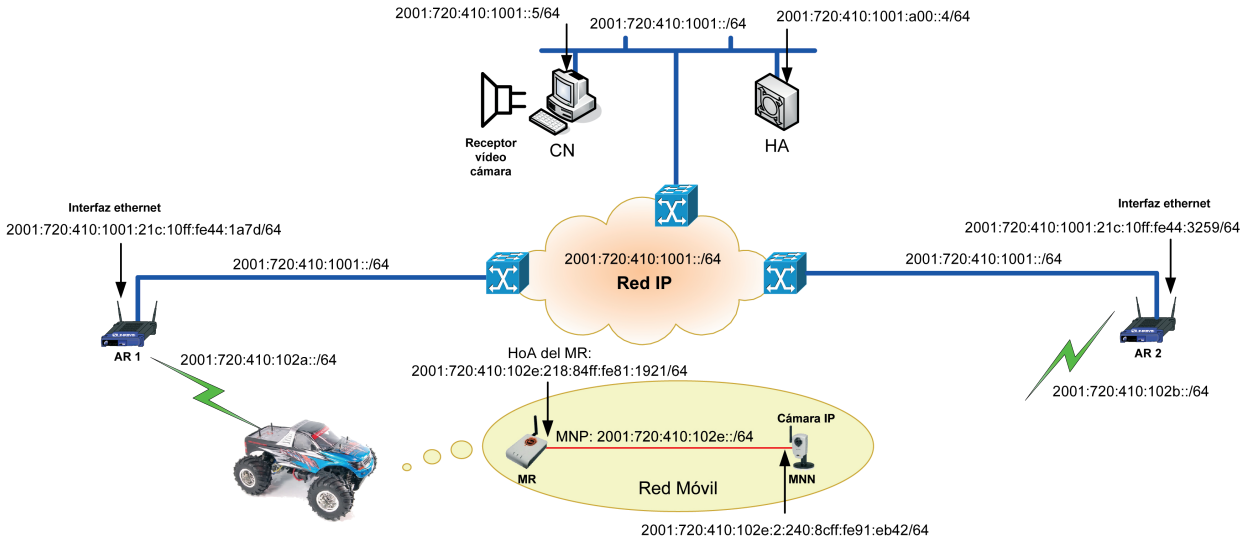

Figura 5.3: Escenario de prueba completo.

5.3. Validación del prototipo

En primer lugar se procede a validar el prototipo únicamente con un router móvil, siguiendo el escenario marcado en la Figura 5.1. Para ello se ejecuta el *software* en ambos dispositivos, comprobando que el nodo móvil se conecta a un router de acceso, en este caso en el que está en el canal 4, a partir de la recepción de un *router advertisement*.

A continuación el router móvil configura su dirección de la interfaz *egress* a partir de dicho anuncio, según el proceso comentado en el capítulo 2. Una vez hecho esto, envía al agente local un mensaje de señalización, *Binding Update* (ver sección B.2), y éste le responde con un mensaje de asentimiento, *Binding ACK* (ver sección B.3). Dicho intercambio de mensajes se muestra en la Figura 5.4, con una captura del programa *Wireshark*.

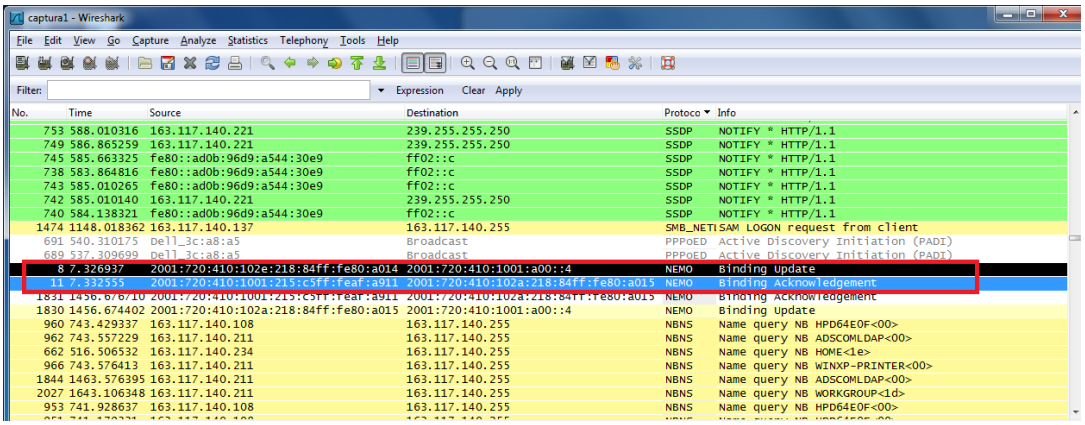

Figura 5.4: Intercambio de mensajes de señalización del protocolo NEMO BS.

En las Figuras 5.5 y 5.6 se muestran en detalle los mensajes BU y BA respectivamente. Estos mensajes se comentan más detalladamente en el Apéndice B.

En la Figura 5.5, correspondiente al *Binding Update*, mensaje enviado por el MR al HA para indicarle su punto de acceso tan pronto como el MR haya adquirido su Care-of Address al recibir el router advertisement procedente del router de acceso de la red visitada, se pueden observar los campos típicos de este mensaje de señalización tales como la cabecera de movilidad. Es una cabecera de extensión usada por los routers móviles y agentes locales en los mensajes referentes a la creación y gestión de las asociaciones de movilidad. Esta cabecera se indica dentro del campo *Next Header* de la cabecera precedente por un valor de 135 (0x87). Dentro de esta cabecera se puede destacar el valor del campo *MH Type*, que identifica el tipo de mensaje de movilidad. En este caso su valor es 5 correspondiente al del mensaje BU. Ya dentro del mensaje BU señalar los bits *Home Registration* y *Mobile Router Flag*). El primer flag indica al receptor del mensaje que actúe como agente local del MR y el segundo indica al agente local si el BU proviene de un nodo o un router móvil. Si tiene valor cero el agente local supondrá que el router móvil está funcionando como nodo y no le reenviará los paquetes destinados a la red móvil.

Otras opciones que se encuentran en este mensaje son:

- *Alternate Care-of Address*: Contiene la dirección que se usará como *Care-of Address* en el proceso de registro en el agente local, en lugar de utilizar la dirección origen del paquete IPv6.
- *Home Address Option* Es utilizado por el nodo móvil mientras está fuera de casa, para informar al agente local de su dirección en la red hogar.

En la Figura 5.6 se muestra el mensaje correspondiente al *Binding ACK*, enviado por el HA al MR como asentimiento del BU. Algunos campos que cabe mencionar son:

- *Status*, que indica si el BU es aceptado o no. En este caso su valor es cero, por lo tanto, ha sido aceptado.
- *Mobile Router Flag*: Indica que el agente local que procesó el mensaje *Binding Update* soporta routers móviles. En la captura se comprueba que está a uno por lo que el HA ha configurado el encaminamiento de tráfico hacia la red móvil.

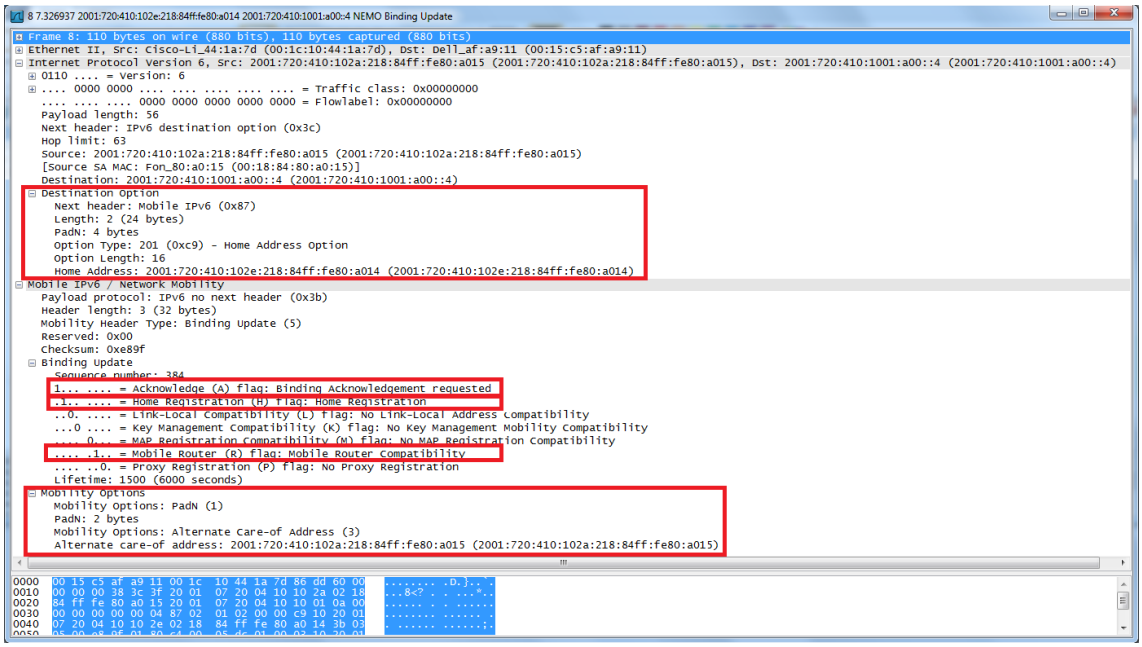

Figura 5.5: Vista detalla del mensaje BU.

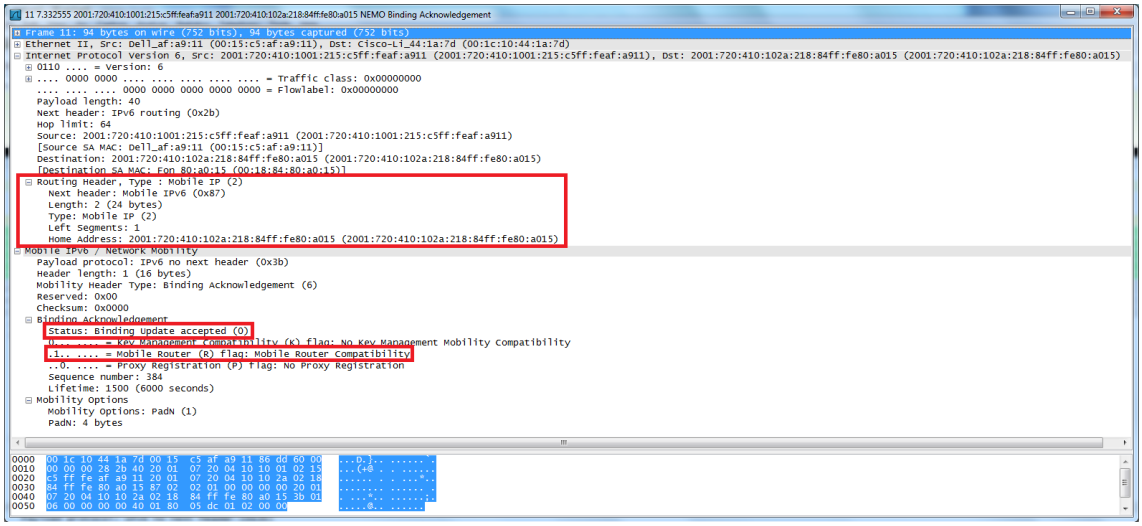

Figura 5.6: Vista detalla del mensaje BA.

Tras el intercambio de mensajes entre ambos equipos, tanto el agente local como el router móvil, establecen un túnel IPv6 del que serán sus extremos. En las Figuras 5.7, 5.8 y 5.9 se muestran las salidas por pantalla de los terminales para ambos nodos, con su correspondiente configuración, intercambio de mensajes y creación de túneles. El nivel de detalle de mensajes de la aplicación está en modo depuración, por lo que es mayor la información obtenida y así conseguir un mejor entendimiento de la ejecución del mismo. En la Figura 5.7 se observa como el agente local en primer lugar lee su fichero de configuración e inicializa el valor de sus routers móviles y prefijos MNPs correspondientes. Tras esto se observa como se queda a la espera de la recepción de mensajes BU. Al recibirlo, realiza las comprobaciones pertinentes y tras esto, las acciones que considere oportunas en la *Binding Cache* y si es necesario, levanta el túnel, quedándose a la espera de nuevo de mensajes *Binding Update*.

El flujo de programa que sigue dicha entidad se comenta con mayor nivel de detalle en la subsección 3.3.2.

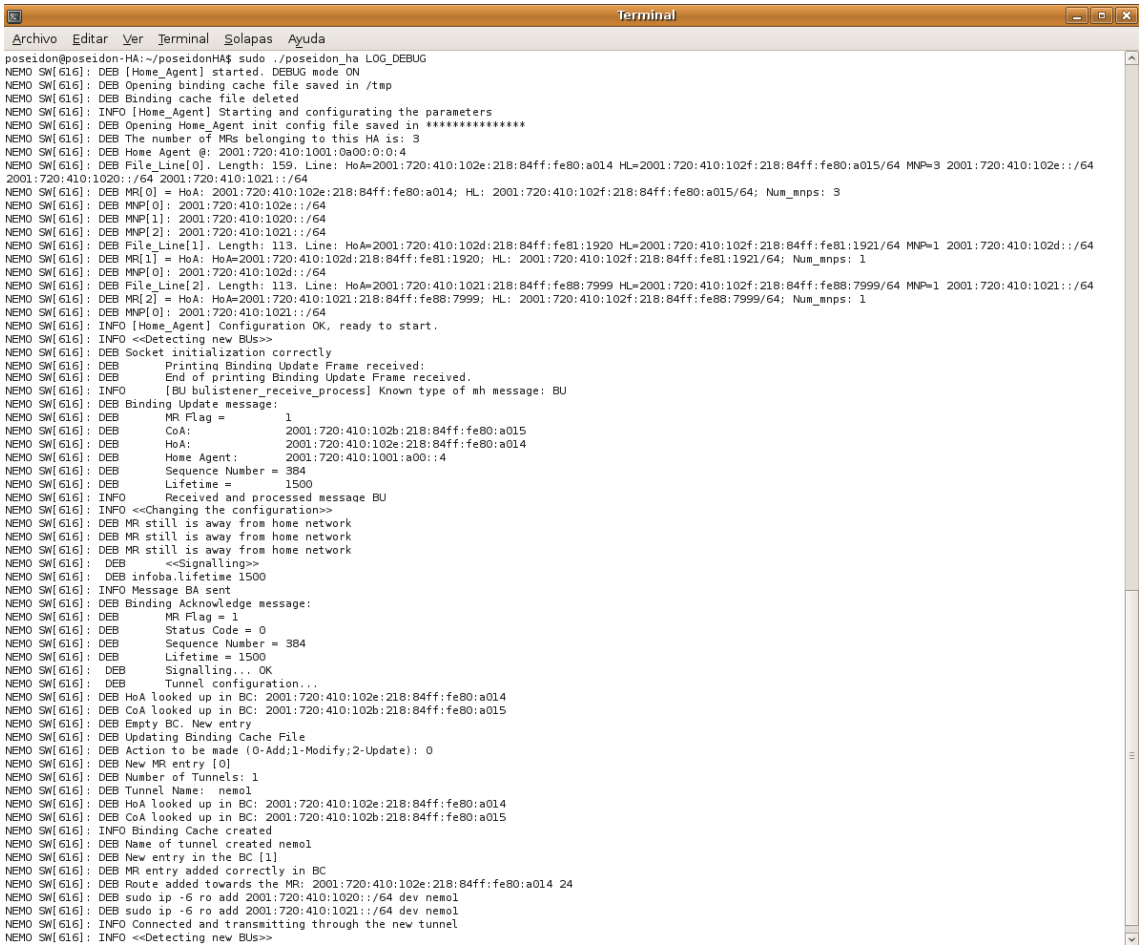

Figura 5.7: Ejecución del *software* en el agente local.

En las Figuras 5.8 y 5.9 se observa como el router móvil en primer lugar también lee su fichero de configuración e inicializa el valor de sus MNPs, dirección del agente local, dirección que toma en la red hogar, etc. Tras esto se observa como se queda a la espera de la recepción de mensajes BU. Al recibirlo, realiza las comprobaciones pertinentes y tras esto, las acciones que considere oportunas en la *Binding Cache* y si es necesario, levanta el túnel, quedándose a la espera de nuevo de mensajes BU.

El flujo de programa que sigue dicha entidad se comenta con mayor nivel de detalle en la subsección 3.3.3.

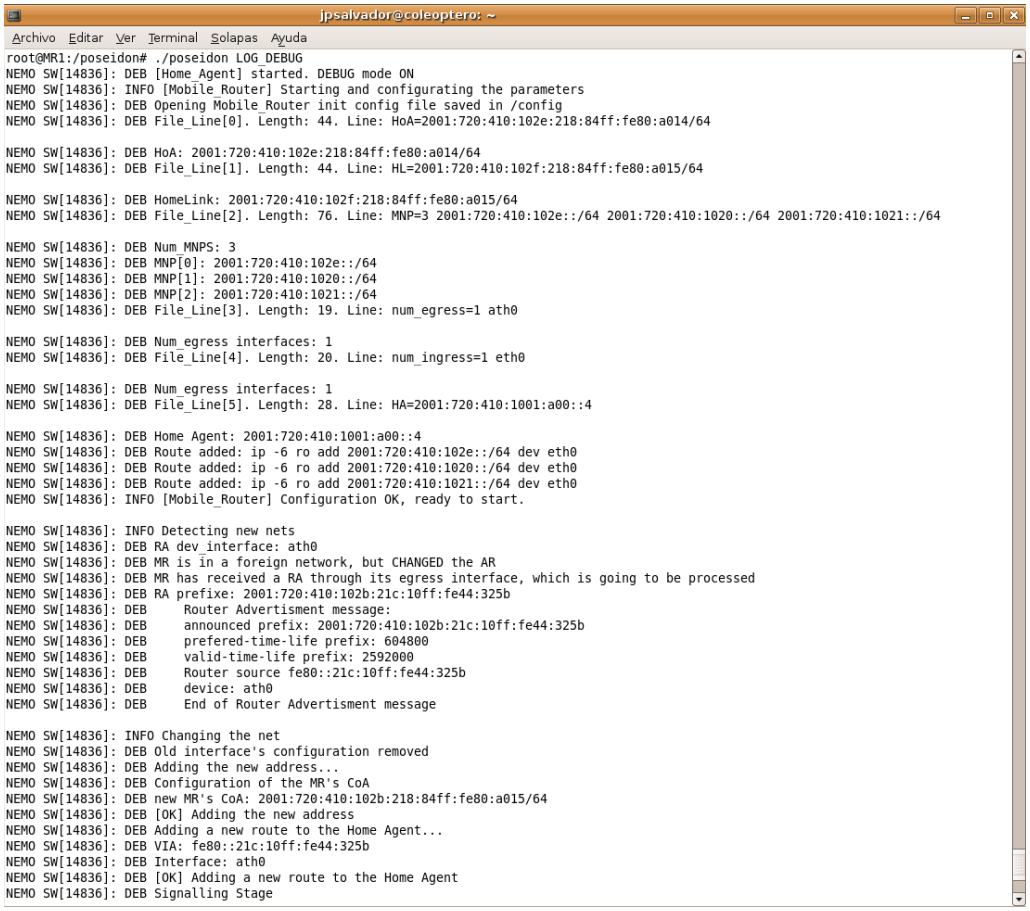

Figura 5.8: Ejecución del *software* en el router móvil (I).

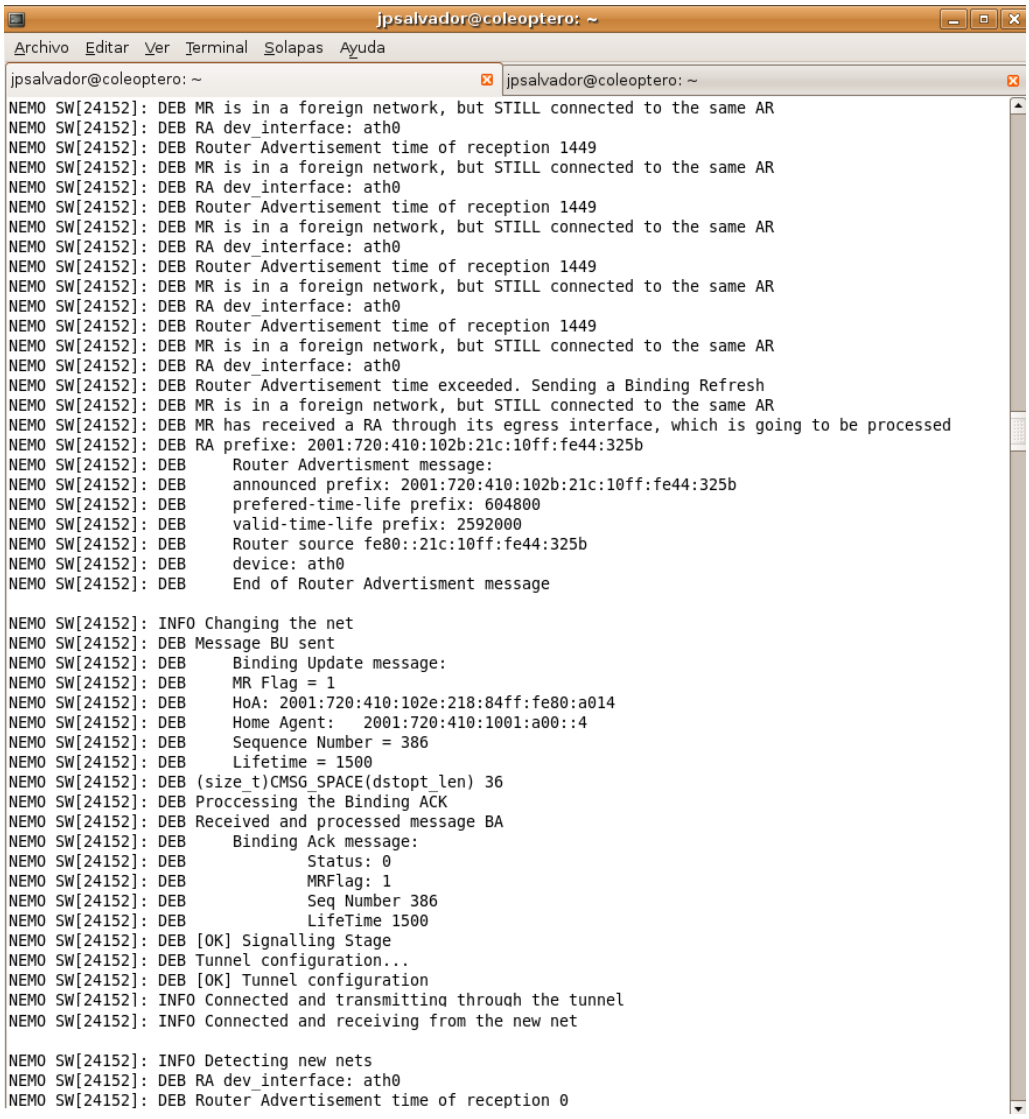

Figura 5.9: Ejecución del *software* en el router móvil (II).

A continuación con la ayuda del comando *ping6* se comprueba la interoperabiliadad del *software*. Se envían paquetes ICMP desde el nodo corresponsal al nodo móvil. En la Figura 5.10 se puede comprobar que existe comunicación entre ambas entidades. Ahora queda comprobar que el túnel se ha levantado correctamente en ambos extremos y que la comunicación se produce a través del mismo. Para esta comprobación se utiliza el programa *tcpdump*[1], capturando en el router móvil en la interfaz perteneciente al túnel (en el ejemplo *nemo1*). En la Figura 5.11 se comprueba como el tráfico generado por la aplicación ping es enviado a través del túnel.

Después se captura el tráfico en la interfaz inalámbrica del router móvil. Esto va a permitir confirmar que el tráfico que va a través del túnel, una vez que se ha establecido previamente, es encapsulado en cabeceras IPv6 como ya se ha explicado anteriormente en la sección 2.3.3. Si se captura en la interfaz del túnel (*nemo1*) las capturas no poseen la encapsulación IPv6, ya que ésta se realiza en los extremos del túnel.

[1] http://www.tcpdump.org/

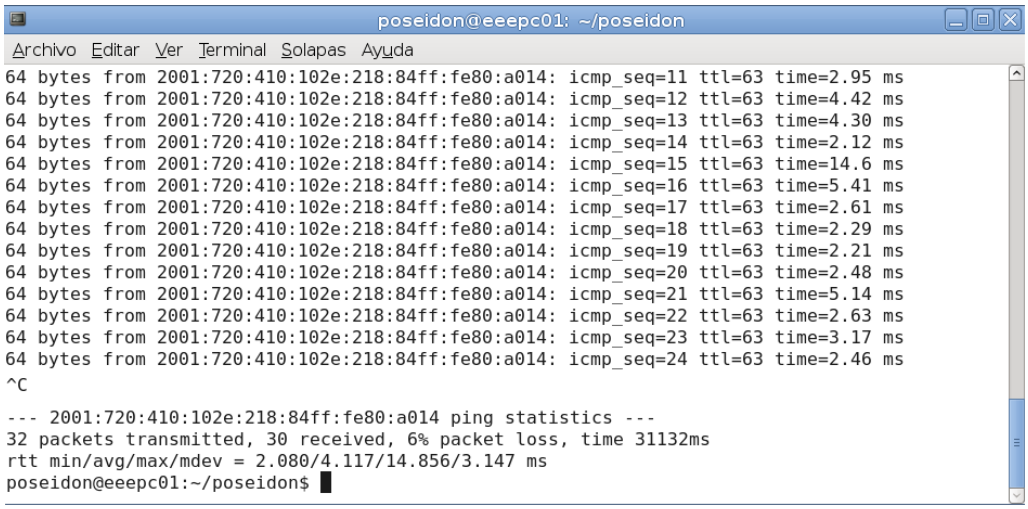

Figura 5.10: Comprobación de comunicación entre HA y MR.

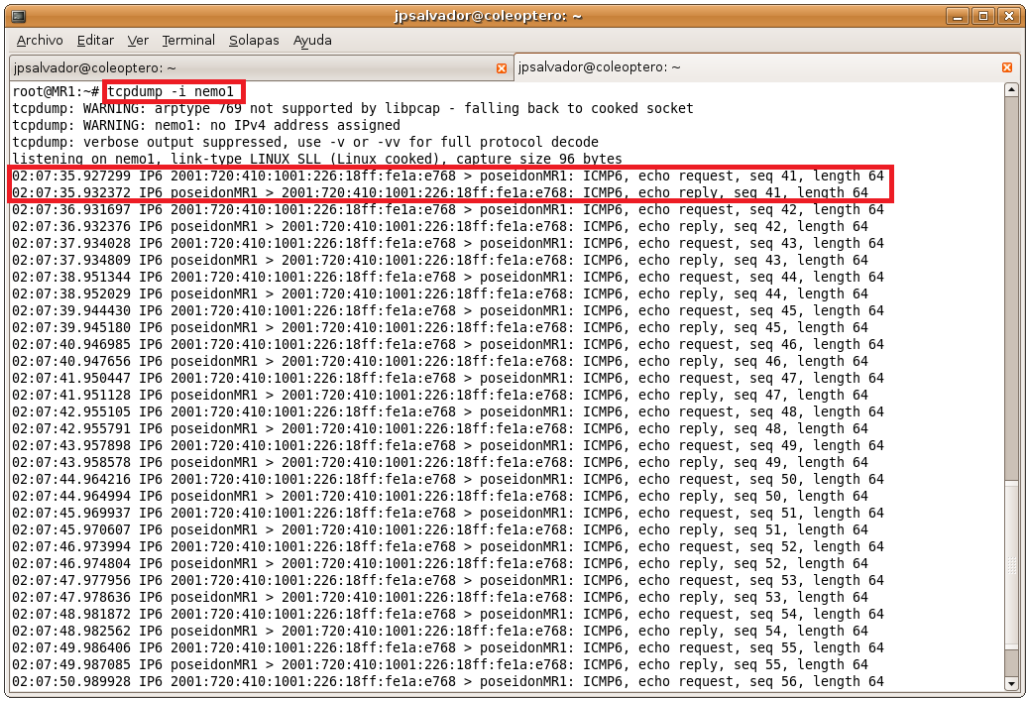

Figura 5.11: Comprobación de comunicación entre HA y MR a través del túnel.

En este caso, como se puede comprobar al ver las Figuras 5.12 y 5.13, dicha cabecera la establece el HA en el mensaje *echo request*, que redirige el trafico procedente del nodo corresponsal hacia el router móvil. En el posterior mensaje, *echo reply*, es el router móvil el encargado de añadir la cabecera IPv6 para que el tráfico sea redirigido por el túnel, siendo el agente local en el que en este caso la suprima, reenviando el mensaje al nodo corresponsal a través de la red.

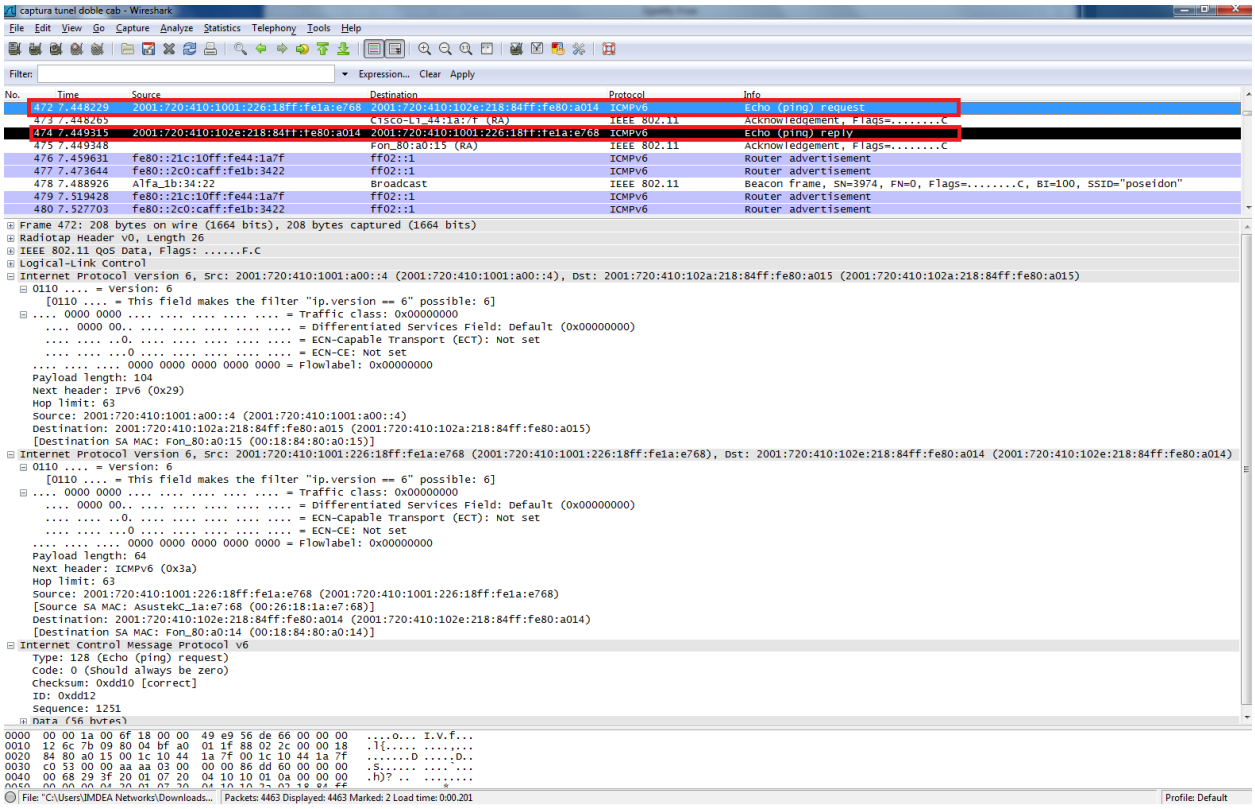

Figura 5.12: Encapsulación de los paquetes que van a través del túnel.

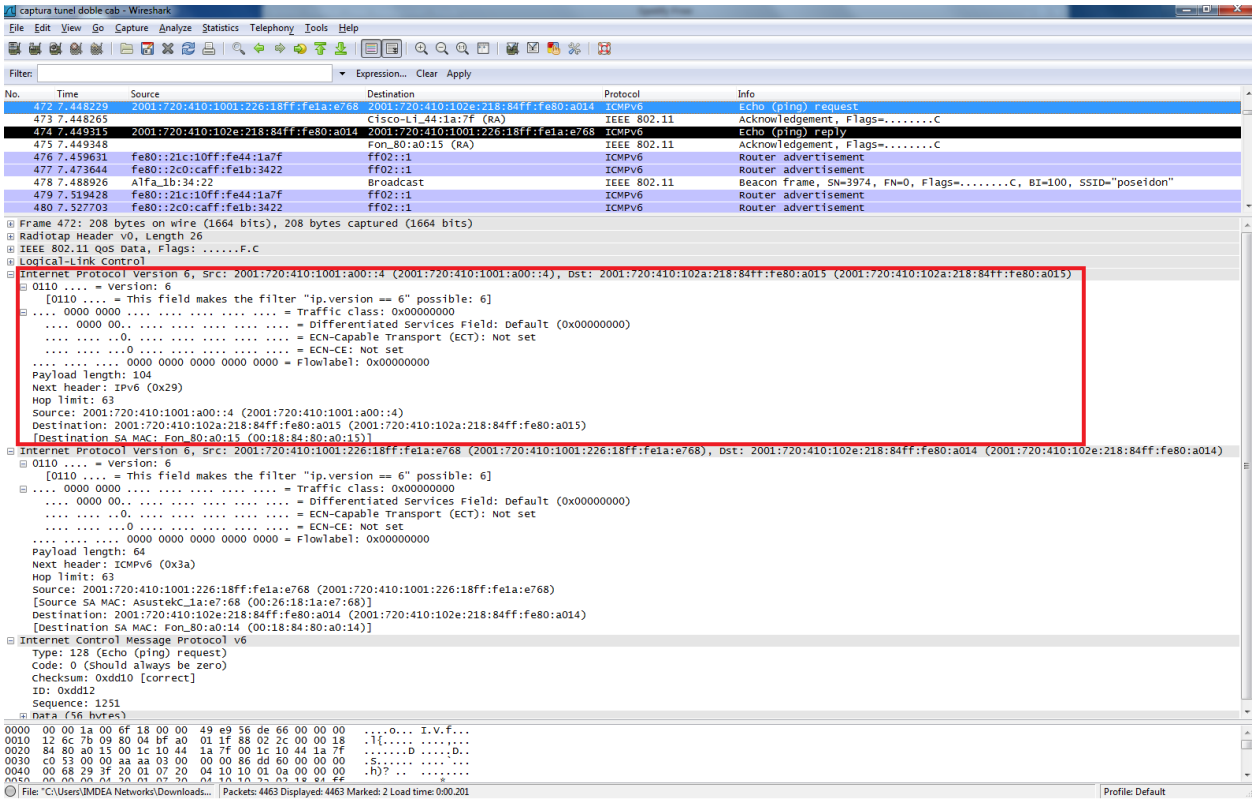

Figura 5.13: Vista detallada de la cabecera de encapsulación.

Una vez que se ha validado el funcionamiento básico del protocolo, se realizan una serie de pruebas o *tests* para comprobar el correcto funcionamiento de la aplicación con mayor detalle. La primera de ellas va a consistir en un envío mayor de datos usando el protocolo TCP, que presenta mayor complejidad que el protocolo UDP como se comenta en la sección H.2.

Con esta prueba se pretende demostrar la viabilidad de la aplicación, ya que si la espera de retransmisiones resulta elevada la comunicación se perderá. Para ello, se va a descargar desde el nodo corresponsal un fichero situado en el nodo móvil haciendo uso del protocolo SCP[2] que utiliza como protocolo de transporte TCP.

```
scp -6 -v root@[2001:720:410:102e:218:84ff:fe80:a014]:/tmp/Kaboom.mp3 .
```

Como se puede ver en las Figura 5.14 se establece la conexión, comprobándose los parámetros necesarios entre cliente y servidor. En ese momento se simula el movimiento de la red móvil forzando un cambio de punto de acceso y los paquetes que se han perdido se retransmiten de nuevo. Finalmente en la Figura 5.15 se observa como la descarga termina con éxito una vez que se han recibido todos los paquetes pendientes.

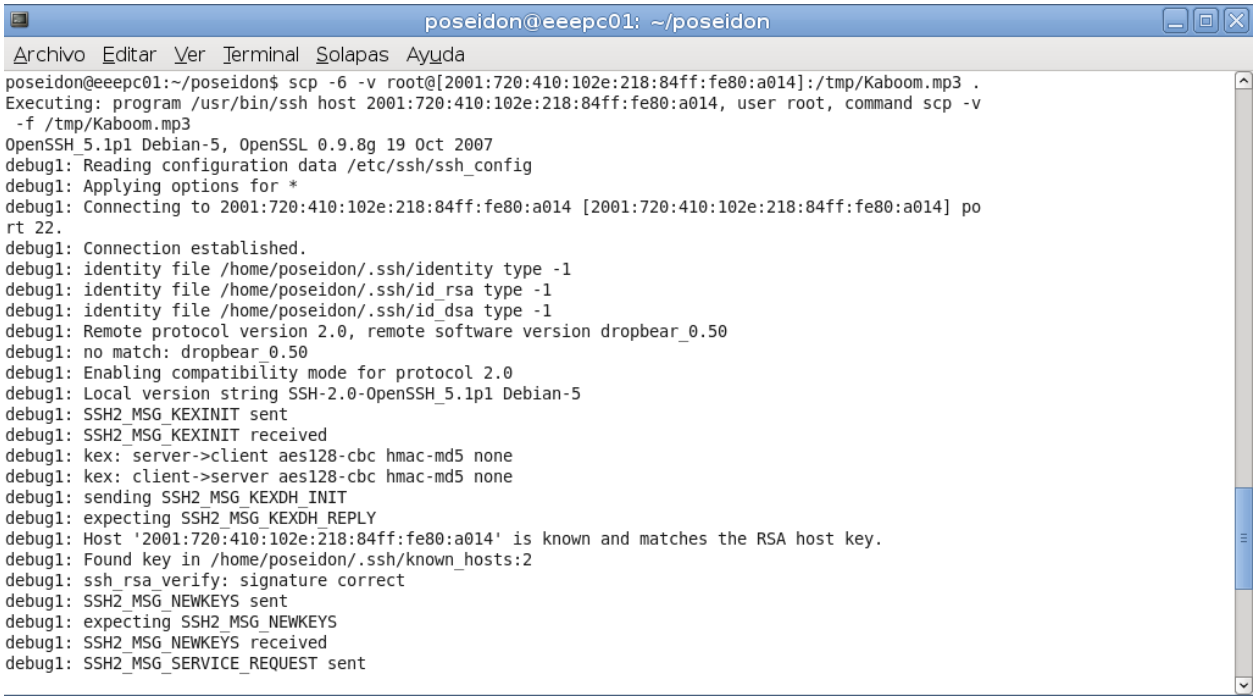

Figura 5.14: Captura del inicio de la descarga

[2] ***S**ecure **C**opy **P**rotocol*:Protocolo que permite la transferencia de ficheros. Utiliza el puerto 22, se basa en el protocolo RCP de BSD que es tunelizado a través del protocolo SSH para proveer encriptación y autenticación. Puede incluso no considerarse un protocolo en sí, sino más bien como una combinación de RCP y SSH.

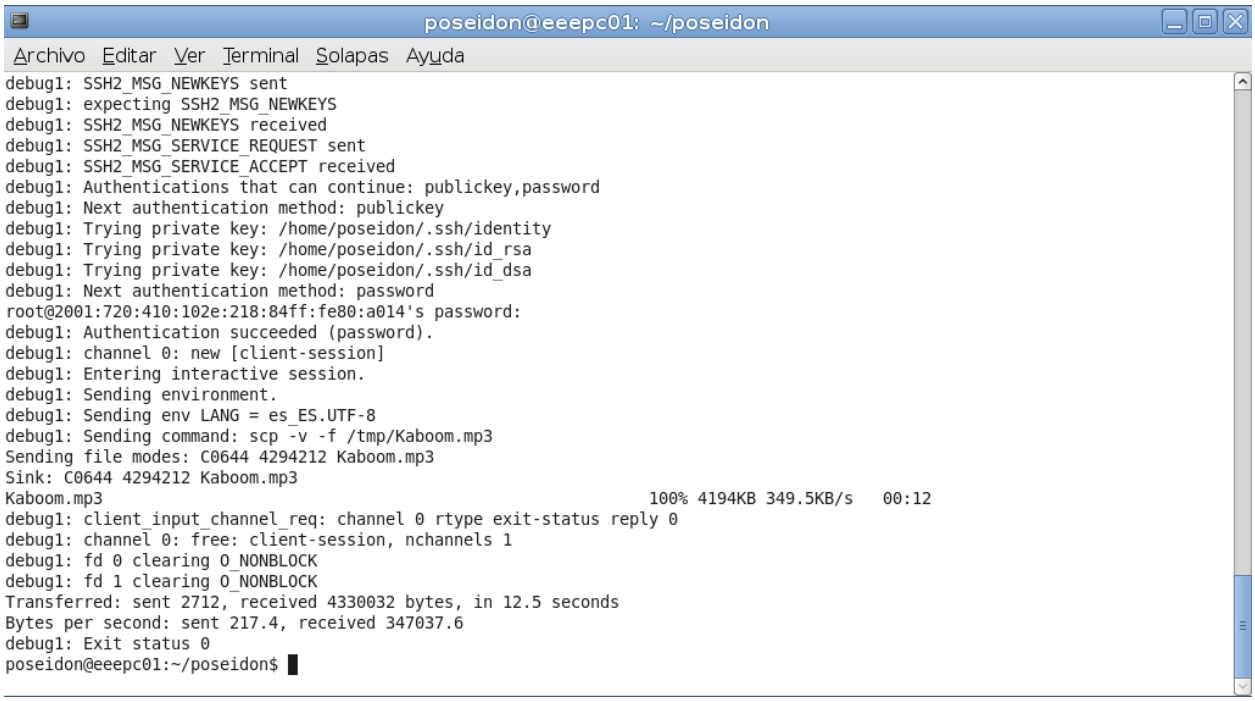

Figura 5.15: Captura del final de la descarga.

Con la prueba anterior se demuestra que el protocolo es transparente para los niveles superiores, ya que los extremos finales de la comunicación no son conscientes del movimiento, tan sólo notarían una pequeña perdida de datos cuando se produce el traspaso. Sin embargo, al utilizar TCP, que proporciona mecanismos para proteger la comunicación, esta pérdida no supone un grave problema, ya que los paquetes que no lleguen al destino serán retransmitidos. No obstante, si el tiempo empleado en el traspaso fuera muy elevado, llegaría un momento en el que los temporizadores asociados a la retransmisión de TCP obligarían a cerrar la conexión, siendo necesario establecer una nueva para poder realizar la descarga. En el caso de utilizar otro protocolo a nivel de transporte que no proporcionara un servicio fiable, como UDP, sería la aplicación la que debería contar con algún mecanismo de recuperación. Por eso, las aplicaciones para las que la pérdida de algún paquete no es crítica utilizan UDP, como por ejemplo, transmisiones de voz o vídeo (ver Apéndice H).

La siguiente de las pruebas va a consistir en la validación del prototipo para varios routers móviles. Para ello se va a añadir un nuevo nodo móvil al escenario planteado en la Figura 5.1 y se procede de igual forma que en el caso anterior, pero con la única diferencia que ahora se cuenta con un nodo más. En la Figura 5.16 se comprueba como el agente local crea un nuevo túnel (*nemo2*) tras producirse la correcta señalización entre ambas entidades.

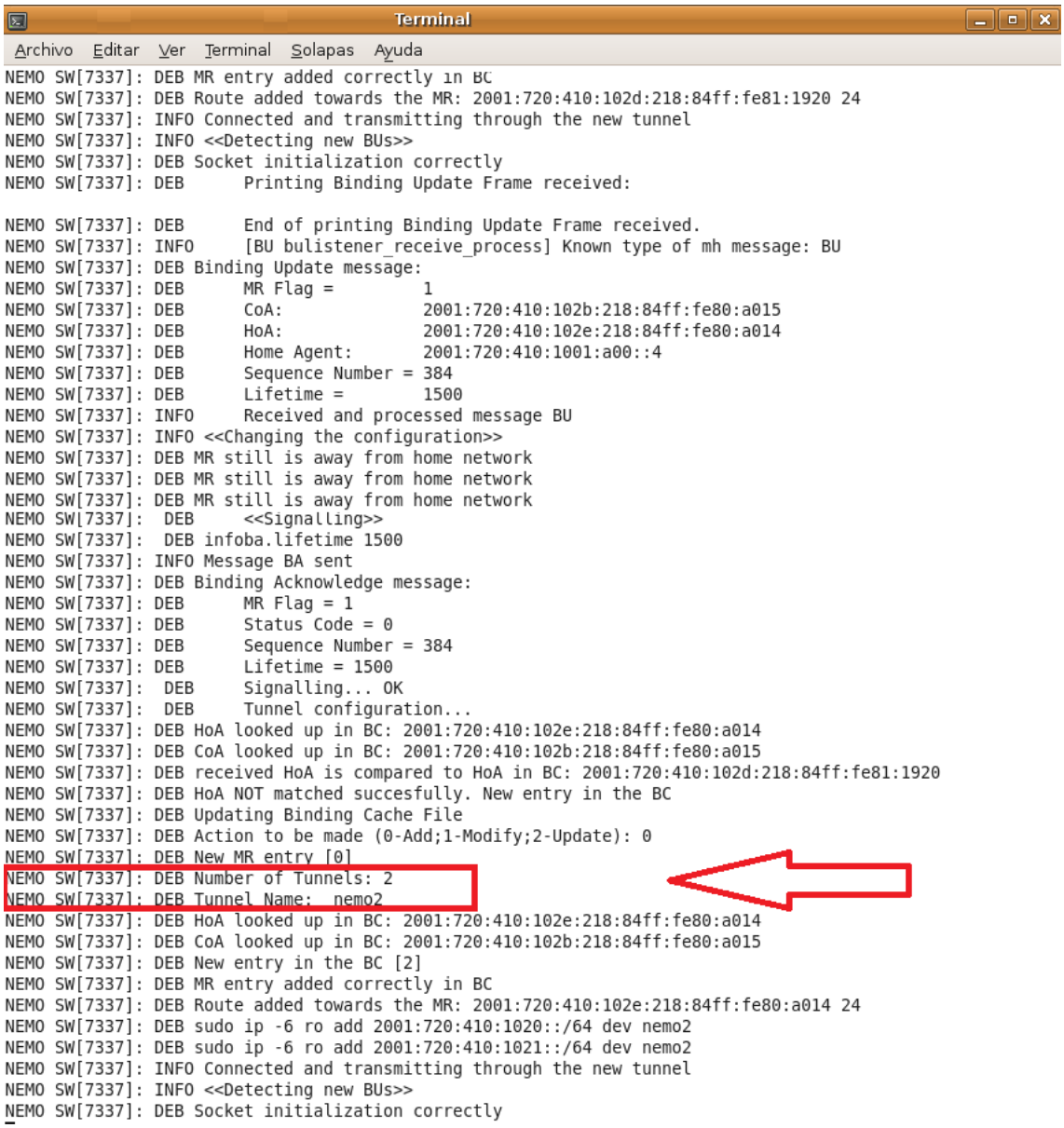

Figura 5.16: Ejecución del *software* en el agente local para varios routers móviles.

En último lugar, se va a simular el movimiento de la red forzando el cambio de red visitada en el router móvil. Para comprobar que el túnel se mantiene y no se pierde la comunicación se envían paquetes ICMP (tamaño de datos = 52 bytes) mediante el programa ping6 (Figura 5.17) cada 250 ms.

En la Figura 5.10 se marca mediante un recuadro el momento en el que se ha producido el traspaso. Se observa que cuando se cambia de punto de acceso se pierden una serie de paquetes, pero la comunicación se recupera rápidamente entre el nodo corresponsal y el nodo móvil.

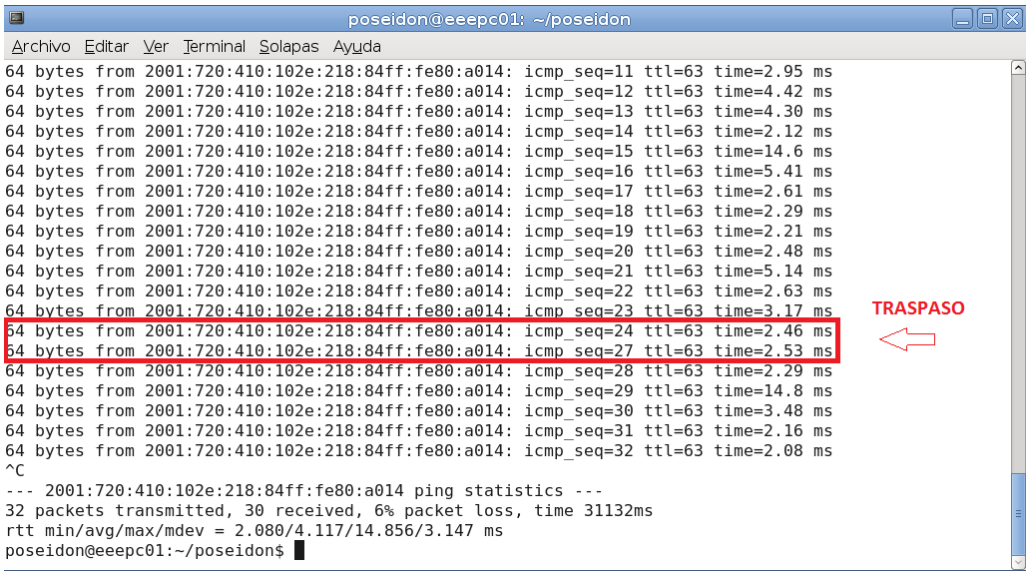

Figura 5.17: Validación de cambio de red visitada.

5.4. Pruebas de estabilidad

Otra de las pruebas realizadas para validar el prototipo, han sido pruebas de estabilidad de la aplicación. Se han realizado pruebas forzando el cambio de punto de acceso cada minuto, y a su vez, utilizando el comando *ping* para realizar envíos de paquete ICMP cada segundo. El tamaño del paquete empleado era de 52 bytes. Se obtuvo un resultado de:

$$Estabilidad = 53h\ 12min\ 50segundos \tag{5.1}$$

Sin embargo la finalización de la aplicación no fue debido a un fallo del *software*, sino que fue debido al planificador del sistema operativo. A pesar de esto, la limitación de tiempo en un escenario real va a venir dada por la autonomía del router Fonera, como se puede comprobar en los resultados mostrados en la Tabla 5.4.

En los siguientes apartados se van a describir las pruebas realizadas para determinar el rendimiento del protocolo básico de movilidad de redes NEMO mediante el análisis de diversos parámetros tales como el tiempo de traspaso o la tasa obtenida.

5.5. Pruebas de traspaso

Esta primera prueba consiste en evaluar el tiempo de traspaso o *handover* cuando el router móvil cambia de punto de acceso. Este traspaso se produce debido a que la calidad del enlace disminuye por la causa que sea, por ejemplo por el movimiento de la red móvil, cambiando dicho router de punto de acceso (*hard handover*). Durante este tiempo los clientes de la red móvil se encuentran sin conexión a Internet, por lo que se hace deseable que sean lo más breve posible, principalmente en aplicaciones relacionadas con el *streaming* de vídeo o voz. El tipo de handover de acuerdo a la tecnología utilizada es *horizontal* ya que el cambio de punto de acceso se realiza usando la misma interfaz de red, es decir, la misma tecnología.

Para la realización de esta prueba se toman 30 muestras. La prueba consiste en realizar desde el nodo corresponsal *ping* a un nodo móvil con paquetes ICMP de 52 bytes de datos cada 10 ms. Para calcular este tiempo de traspaso se comprobará el número de paquetes perdidos y se multiplicará por el periodo de transmisión:

$$Tiempo\ de\ traspaso = Num\_Paquetes\_Perdidos * Periodo\_TX \tag{5.2}$$

En la Tabla 5.1 se muestran los resultados obtenidos.

Canales	**Tiempo de traspaso [s]**	
	Media	*Desviación típica*
Canales 4 y 5	1.515	0.350
Canales 4 y 11	1.106	0.318

Tabla 5.1: Tabla comparativa de las pruebas de tiempo de traspaso con un router móvil

Los resultados de la Tabla 5.1 muestran un valor elevado en el tiempo de traspaso. A continuación se explicará el porqué de este valor. Nótese la diferencia de los resultados cuando los canales son contiguos (canales 4 - 2.427 GHz - y 5 - 2.432 GHz -) y cuando se encuentran separados o no adyacentes (canales 4 - 2.427 GHz - y 11 - 2.462 GHz -). En el primer caso, los resultados obtenidos son peores que cuando los canales se encuentran separados. Este resultado es coherente porque cuando los canales se encuentran juntos, se produce un solapamiento y, por tanto, interferencias, provocando pérdidas y un aumento del tiempo de traspaso.

Cabe destacar que para la realización de la prueba se desactiva el mecanismo de Detección de Direcciones Duplicadas, DAD (*Duplicate Address Detection*) 5.4 de [TN98] con la intención de que el retardo medido no se vea influenciado por este proceso. DAD se ejecuta antes de asignar una dirección a una interfaz para prevenir que múltiples nodos usen la misma dirección simultáneamente, mediante el intercambio de mensajes *Neighbour Solicitation* y *Neighbour Advertisement*, lo cual añade un retardo variable que aumenta el periodo de latencia y dificulta la evaluación del impacto del prototipo en el tiempo de traspaso.

Dado que el tiempo de traspaso debe ser lo menor posible, se intenta reducirlo haciendo la implementación más eficiente, pero finalmente se determina que para reducirlo más es necesario cambiar o modificar el controlador ó *driver* de la tarjeta inalámbrica de los routers Fonera, utilizados como router móvil, ya que el traspaso a nivel 2 no es eficiente.

Otras soluciones, en el caso de tener un router con varias interfaces *egress*, serían por ejemplo, usar un algoritmo de escaneo selectivo basado en un mecanismo de caché [For04] permitiendo que el MR antes de realizar el cambio de punto de acceso escuche a varios ARs al mismo tiempo, por lo que se reduce la duración del traspaso (*soft handover*). El problema de ésto es que durante el periodo de traspaso los paquetes llegan duplicados, ya que el MR está conectado a ambos puntos de acceso.

Algunas de las comprobaciones que llevan a la conclusión de que para reducir este tiempo es necesario modificar el controlador de la tarjeta o realizar otro tipo de traspaso fueron:

- Que el que envía el tráfico sea un nodo móvil en lugar del propio MR, esto además se

aproxima más a un escenario real, en el que el MR actúa como *forwarder* del tráfico de su red móvil. El rendimiento en este caso fue algo mejor (pruebas de rendimiento descritas posteriormente), lo que puede dar una idea de la limitación de recursos del router Fonera.

- Descargar un archivo mediante SCP con o sin traspaso. Esta prueba confirmó que el tiempo medio en el que se realiza un cambio de acceso está alrededor de 1 segundo.

- Medir con un analizador de redes el tiempo que dura el proceso de señalización del protocolo cuando se produce un traspaso. Este tiempo resulta en unos pocos milisegundos (*20ms*) desde que el router móvil envía el mensaje BU y recibe el asentimiento del mismo. También se comprueba el valor de esta señalización mediante el uso de la estructura *timeval time* de C y la función *gettimeofday()*, obteniendo de igual forma 20ms.

Como ya se ha comentado anteriormente el traspaso a nivel de enlace no es eficiente. Para demostrarlo se va a descomponer el tiempo de traspaso en sus fases más significativas:

$$T_{TRAPASO} = T_{TRAPASO\_L2} + T_{MD} + T_{SIGNALLING} + T_{CONF} \tag{5.3}$$

El tiempo de configuración es el tiempo empleado por el MR y el HA en configurar las rutas y túneles correspondientes. Este tiempo se considera despreciable frente al resto.

El tiempo de señalización $T_{SIGNALLING}$ es el tiempo transcurrido desde el envío del BU hasta la recepción del BA. Este tiempo se ha medido varias veces (20 repeticiones) con la ayuda de un analizador de redes (*tcpdump*), obteniéndose una media de **30 ms** .

A continuación el tiempo de detección de movimiento T_{MD} , que es el tiempo que tarda el MR desde que se asocia con el nuevo punto de acceso hasta que envía el mensaje de señalización BU. Se realizan 20 repeticiones, esperando tener suficiente variedad muestral para dar rigor a los resultados, obteniendo: **0.376 s**.

Por último, el tiempo de traspaso a nivel dos es el factor dominante en esta figura de mérito. Este tiempo es el transcurrido desde la recepción del RA por parte del router móvil hasta que se asocia con el AP. Es a nivel de enlace y durante ese tiempo tiene lugar la autenticación y asociación del MR con el nuevo punto de acceso. El tiempo de traspaso a nivel dos es de: **0.70 s**.

En la Figura 5.18 se observan la mayoría de los mensajes involucrados en este proceso

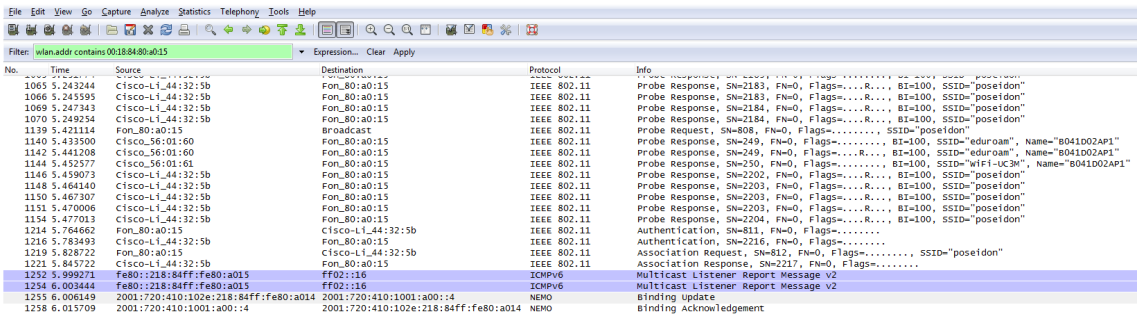

Figura 5.18: Captura de los mensajes intercambiados durante el proceso de traspaso.

5.6. Pruebas de tasa de envío

Otra de las pruebas con la que se pretende evaluar el rendimiento de nuestro prototipo es a través de la eficiencia o *throughput* para distintos tipos de tráfico. Concretamente, se ha considerado el envío de paquetes UDP a una tasa de 11 Mbps y a diferentes tamaños de paquete, y el envío de tráfico TCP, para 802.11g.

Para la realización de esta prueba se va a enviar tráfico UDP y TCP entre el nodo móvil (MNN) y el CN. Esto es debido a que el hecho de que los routers móviles tengan que generar el tráfico ellos mismos, limita su capacidad de procesamiento, además hay que tener en cuenta la limitación de sus recursos, como pueden ser entre otros, el procesador o la memoria RAM. Asimismo en una situación real, sería un usuario o nodo móvil el encargado de generar la mayor parte del tráfico. Para ello se configura un PC como nodo móvil del MR y se conecta dicho nodo con un cable Ethernet al router móvil. A partir del *router advertisement* que envía el MR, el nodo configura una dirección perteneciente a uno de los MNPs gestionados por el MR. El tamaño del campo de datos a nivel UDP (APDU - *Application Protocol Data Unit*) de los datagramas es:

$$\boldsymbol{Payload} = 1500B(MTU) - 8B(UDP) - 40B(IPv6) = \mathbf{1452\ B} \tag{5.4}$$

$$\boldsymbol{Payload\_NEMO} = 1500B(MTU) - 8B(UDP) - 40B(IPv6) - 40B(IPv6) = \mathbf{1412\ B} \tag{5.5}$$

Por lo tanto las pruebas que se van a realizar son:

1. UDP a 11 Mbps (sin fragmentar). Si se usa NEMO los datos efectivos como se comprueba en la expresión anterior son 1412 bytes, mientras que en caso de que no se utilice el prototipo los datos útiles serán 1452 bytes.

2. UDP a 11 Mbps (con fragmentación). Para que se produzca fragmentación basta con un añadir un byte más a los valores de la prueba anterior. En el caso de usar NEMO se transmiten 1413 bytes y por contra, sin emplear el protocolo, se transmiten datagramas de 1453 bytes.

3. UDP a 11 Mbps (con paquetes pequeños) Para ambos casos se transmiten paquetes de 100 bytes.

Con ello se pretende evaluar la pérdida de rendimiento introducida por el hecho de tener ejecutando el protocolo NEMO. Idealmente, la pérdida de eficiencia debe corresponderse con la diferencia del tamaño de los datos de control ya que NEMO introduce un *overhead* de 40 bytes, correspondiente a la cabecera que se introduce para redirigir el tráfico a través del túnel IPv6-IPv6. Como nota adicional se verá cómo influye la fragmentación de los datagramas en la tasa efectiva.

En la Tabla 5.2 se muestran los resultados de dicha prueba realizados para 20 iteraciones. Hubiese sido conveniente realizar un mayor número de repeticiones, pero a veces el estado de la red del laboratorio daba problemas y lo único que se conseguía eran resultados que no permitían establecer un modelo coherente para el análisis de esta medida.

Prueba	**Tasa obtenida sin NEMO [Mbps]**		**Tasa obtenida con NEMO [Mbps]**	
	Media	*Desviación típica*	*Media*	*Desviación típica*
UDP A.	10.5142	±0.2534	10.3	±0.2803
UDP B.	7.2873	±0.1694	7.1853	±0.1120
UDP C.	1.2182	±0.0244	1.2020	±0.0094
TCP	6.3150	±0.1622	5.5117	±0.2577

Tabla 5.2: Tabla comparativa de pruebas de rendimiento con un solo router móvil

Una primera aproximación para ver la influencia del *overhead* introducido por NEMO sería a través de las expresiones siguientes:

$$Tasa\ (NEMO) = Payload\ (NEMO)/Payload\ (Sin\_NEMO) * Tasa\ (Sin\_NEMO) \tag{5.6}$$

Por tanto con los datos de la Tabla 5.2 se obtiene:

- Para el caso de no fragmentar y enviar al máximo de datos permitidos:

$$Tasa\ (NEMO) = 1412/1452 * 10.5142 = 10.2245 \tag{5.7}$$

- Para el caso de fragmentar:

$$Tasa\ (NEMO) = 1413/1453 * 7.2873 = 7.0769 \tag{5.8}$$

Por lo tanto, queda demostrado que el funcionamiento del prototipo afecta de la forma esperada, ya que se añaden unos datos de control necesarios para encapsular el tráfico desde/hacia la red móvil.

En cuanto a la fragmentación se observa que al fragmentar los paquetes se introducen un mayor número de datos de control por lo que el *throughput* que se obtiene es bastante menor, ya que por ejemplo por un 1 byte más de datos se introducen en la red, en el caso de NEMO, 96 bytes de control más:

$$Overhead = 8B(UDP)+40B(IPv6)+40B(IPv6)+8B(Fragment\ Header) = 96bytes \tag{5.9}$$

En el caso de paquetes pequeños disminuye de forma considerable la tasa obtenida, ya que el tamaño de los datos de control es comparable al de datos útiles. Además influye el hecho de que los routers tienen que hacer *forwarding* de más paquetes, limitados a su vez por la la capacidad de procesamiento.

En cuanto a TCP la tasa obtenida es menor que para UDP debido a la mayor complejidad del protocolo. Como se puede apreciar la utilización de NEMO no empeora significativamente el rendimiento con respecto al obtenido sin NEMO.

Añadir que en las Figuras 5.20 y 5.19 se demuestra como para el valor de datos teórico calculado para NEMO (1412 bytes) no se fragmenta, mientras que añadiendo un byte más

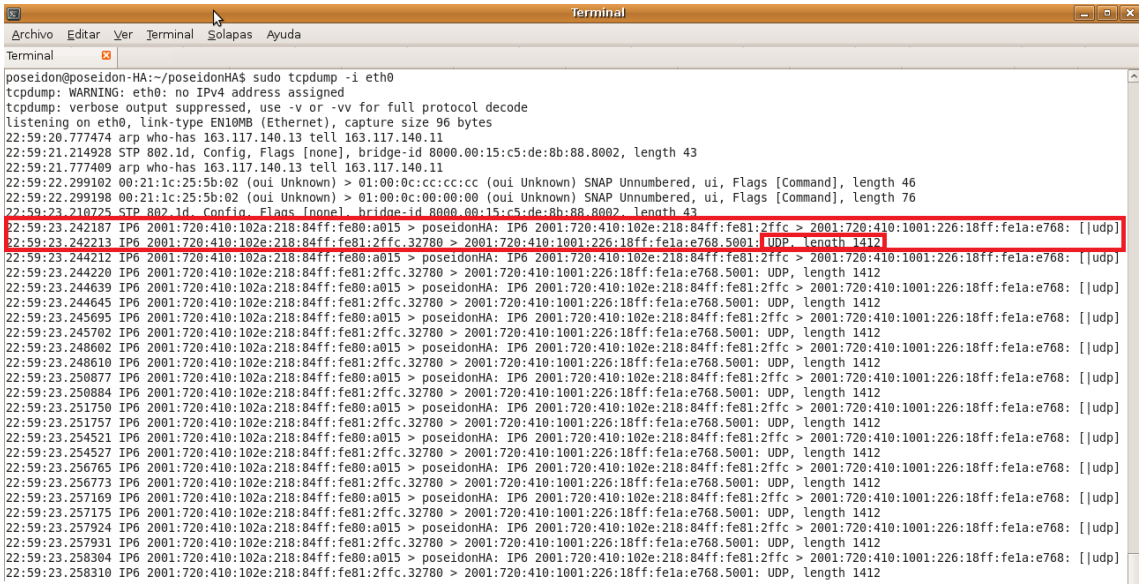

Figura 5.19: Captura de la aplicación *iperf* cuando no se fragmentan los paquetes.

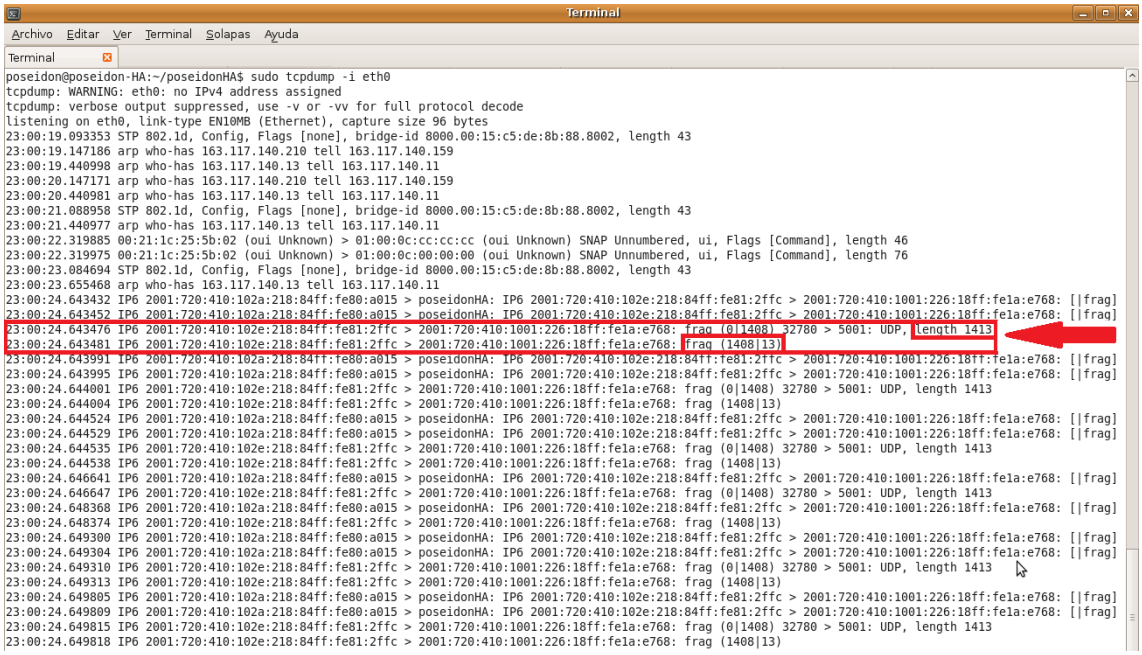

Figura 5.20: Captura de la aplicación *iperf* cuando se fragmentan los paquetes.

(1413 bytes) ya se produce la fragmentación del paquete (IPv6 fragmenta de extremo a extremo).

Para una evaluación más precisa del prototipo también hay que tener en cuenta que en un escenario vehicular real estarán presentes otros elementos, como por ejemplo, más routers móviles. El agente local soporta la gestión de varios MR, de modo que es posible realizar pruebas de rendimiento con más de un MR enviando tráfico. En este caso se añade un nuevo router móvil al escenario. Los resultados se muestran en la Tabla 5.3. Como era de esperar, la tasa media obtenida es menor que en el caso de un sólo MR, pero este descenso no es muy acusado permitiendo todavía una comunicación de cierta calidad. Sin

embargo, se obtenía mucha variación en las medidas según se realizase el movimiento de la red hacia un punto de acceso o hacia el otro, así como dependiendo de qué router móvil se hiciese primero con el medio. De todos modos, el comportamiento del prototipo refleja la problemática asociada a las redes inalámbricas.

Prueba	**Tasa obtenida MR 1 [Mbps]**		**Tasa obtenida MR 2 [Mbps]**	
	Media	*Desviación típica*	*Media*	*Desviación típica*
UDP a 11 Mbps	5.4317	±0.667	5.066	±0.1055
UDP a 35 Mbps	4.90	±0.051	4.878	±0.1683
TCP	4.34	±0.2228	4.516	±0.2141

Tabla 5.3: Tabla comparativa pruebas de rendimiento con dos routers móviles

5.7. Pruebas de consumo de CPU

En este punto se muestran los resultados del consumo de CPU y memoria a la hora de ejecutar la aplicación en los routers móviles. En la Figura 5.21 se observa el efecto que tiene la ejecución del *software* en los dispositivos MR. Cuando esto se produce como se observa en la Figura 5.21 la aplicación consume tan sólo un 1.3 % de la CPU y un 6.5 % de memoria. Por lo que el efecto no es muy notorio. Luego la aplicación hace uso del demonio *syslogd* para gestionar el sistema de mensajes de la aplicación y tener distintos niveles de información para depurar o simplemente tener información acerca de la ejecución del prototipo implementado. El demonio *syslogd*[3] se lanza automáticamente al arrancar un sistema Unix, y es el encargado de guardar informes sobre el funcionamiento de la máquina. Recibe mensajes de las diferentes partes del sistema (núcleo, programas...) y los envía y/o almacena en diferentes localizaciones, tanto locales como remotas, siguiendo un criterio.

En la Figura se muestra que el consumo de CPU para este demonio se ve incrementado al usar la aplicación (8.5 %), sin embargo el uso de memoria es el mismo (2.4 %). En este caso el valor de uso de CPU es bastante mayor, pero esto es debido a que el *software* se ejecuta en modo depuración (*LOG_DEBUG*), si se ejecuta en modo error (*LOG_WARNING*) o en modo información (*LOG_INFO*) el uso de CPU es bastante menor.

[3] *Syslogd*: http://www.syslog.org/

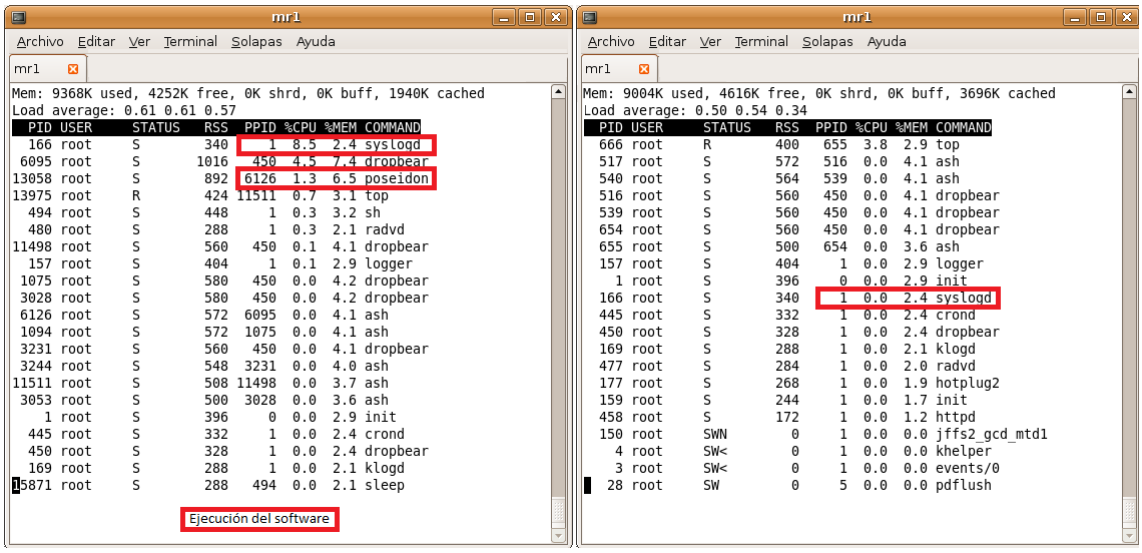

Figura 5.21: Comparación consumo de CPU en el MR al ejecutar la aplicación.

5.8. Pruebas cualitativas

Otra forma de evaluar la implementación realizada es a través de pruebas cualitativas en la que se estima la calidad del vídeo al producirse un traspaso entre diferentes puntos de acceso. Para ello se han escogido a un grupo de sujetos, en nuestro caso 10 personas, que han tenido que rellenar una tabla de acuerdo a las recomendaciones de la ITU [ITU02][ITU96], evaluando la calidad percibida en la transmisión del vídeo en una escala del 1 al 5, siendo 1 para una mala calidad y 5 para una excelente calidad (Figura 5.22).

Five-grade scale			
Quality		**Impairment**	
5	Excellent	5	Imperceptible
4	Good	4	Perceptible, but not annoying
3	Fair	3	Slightly annoying
2	Poor	2	Annoying
1	Bad	1	Very annoying

Figura 5.22: Escalas ITU-R para la calidad y defectos.

En la Figura 5.23 se muestran dichos resultados en términos de calidad media subjetiva con su correspondiente desviación típica. Para evaluar como afecta el traspaso a la transmisión continuada, se realizan varias pruebas con distinta frecuencia de *handover* o traspaso.

Se concluye que a medida que la frecuencia disminuye, la calidad percibida es mayor. Este resultado es esperado, dado que el proceso de handover no está optimizado e introduce una cierta interrupción. Incluso en este entorno, en situaciones donde no se produzcan muchos traspasos, la calidad es percibida como buena por los usuarios.

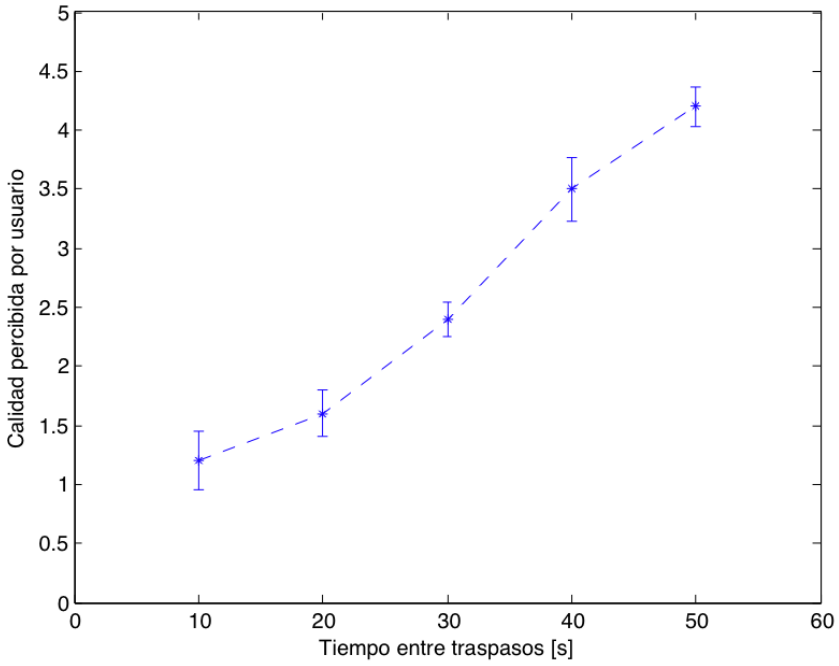

Figura 5.23: Calidad de vídeo percibida por los usuario.

5.9. Pruebas de autonomía del router Fonera

En este proyecto también se han realizado diferentes pruebas de autonomía para medir el consumo de energía en el router Fonera. Se han establecido cuatro posibles escenarios:

1. El primer escenario que se prueba es el router Fonera con su interfaz inalámbrica apagada (Figura 5.24).

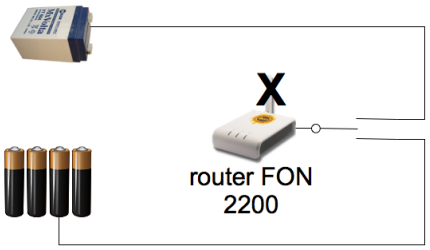

Figura 5.24: Primer escenario de pruebas del consumo de energía en el router Fonera.

2. La siguiente prueba es idéntica a la anterior, excepto que ahora la interfaz inalámbrica está encendida. Además, sin transmitir ningún tipo de tráfico. Para ello se ha configurado la Fonera en modo cliente, y se ha tratado que el canal en el que trabajaba fuera de escaso tráfico (Figura 5.25).

3. En esta tercera prueba se tiene una nueva entidad, un PC, configurado como *Access Point*, el cual envía tráfico hacia el router Fonera. Se han generado paquetes UDP de 1500 Bytes a una tasa controlada sin que haya *overflow* (Figura 5.26).

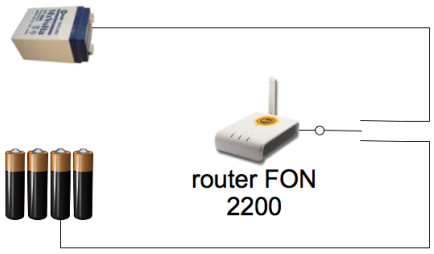

Figura 5.25: Segundo escenario de pruebas del consumo de energía en el router Fonera.

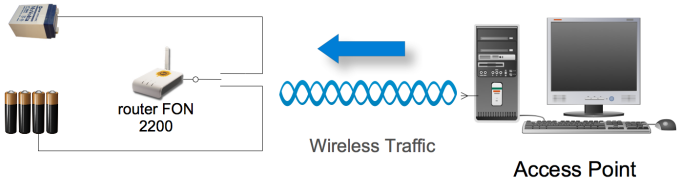

Figura 5.26: Tercer escenario de pruebas del consumo de energía en el router Fonera.

4. En la última prueba, se mantiene el escenario anterior, sin embargo ahora es la Fonera quien genera el tráfico, paquetes UDP de 1500 Bytes, evitando también que haya *overflow* (Figura 5.27).

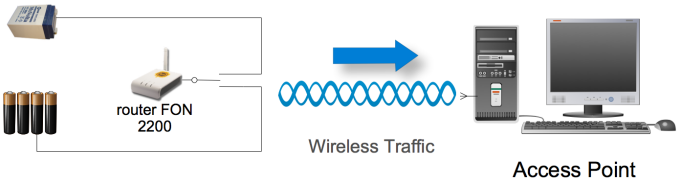

Figura 5.27: Cuarto escenario de pruebas del consumo de energía en el router Fonera.

En la Tabla 5.4 se muestran los resultados de estas pruebas de autonomía. Únicamente se han podido hacer cinco realizaciones en el caso de las pilas y tan sólo tres en el caso de la batería, debido a la larga duración de las pruebas y así no alargar en demasía la etapa de evaluación de la autonomía. Si bien las pruebas no tienen el rigor estadístico que proporcionaría haber hecho más repeticiones, la pequeña varianza de los resultados hace pensar que aumentar el número de repeticiones no hubiese hecho diferir mucho el resultado obtenido en las pruebas.

Se puede comprobar que a mayor tráfico generado por el router Fonera, la duración, indistintamente si son baterías o pilas, es menor. También se puede advertir que lógicamente la duración de la batería es mayor debido a su corriente que es de 4.5A/h, comparada con la de las pilas que es de 2.5A/h.

Finalmente se muestra un gráfico (Figura 5.28) de la comparativa de la duración de la batería para distintos escenarios. Como se puede advertir a mayor tráfico generado por el router Fonera menor duración de la batería.

A modo de resumen, destacar la estabilidad del prototipo realizado que no se interrumpe por fallo del *software*, sino que es el planificador del sistema operativo el que fuerza su finalización. Además, de cumplir todas las especificaciones impuestas por el protocolo

Prueba	**Pilas**		**Batería**	
	Media	*Desviación Típica*	*Media*	*Desviación Típica*
1	5h13min	±15min	18h13m	±10min
2	4h35min	±12min	17h30m	±17min
3	2h56min	±9min	11h17m	±24min
4	2h17min	±16min	9h21m	±26min

Tabla 5.4: Tabla comparativa pruebas de autonomía del router Fonera

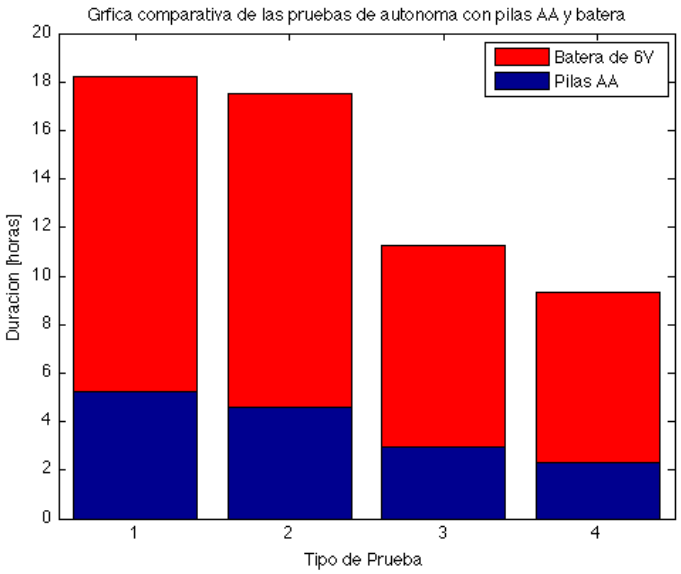

Figura 5.28: Gráfica comparativa de las pruebas de autonomía con pilas AA y batería.

básico de movilidad, se consigue dotar de autonomía a la implementación, asemejándola a una red vehicular. Los resultados muestran que hay algunos aspectos que se pueden mejorar, como es el caso del tiempo de traspaso. Sin embargo, los aspectos de optimización de traspaso no se encontraban entre los objetivos fijados en este proyecto fin de carrera por ello se han desestimado, quedando como propuesta para futuras implementaciones. Otra serie de pruebas si el router móvil contara con más de 1 interfaz, sería la prueba con varios MNPs que, a pesar de eso se ha simulado, comprobando su funcionamiento. Sin embargo, no se realizaron pruebas de rendimiento del mismo ya que la capacidad de gestión del router móvil Fonera imposibilita obtener unos resultados fiables.

En el siguiente capítulo se realizará un resumen del trabajo realizado, enfatizando las puntos a favor y en contra, y recomendando posibles trabajos futuros que permitan extender y mejorar la funcionalidad del prototipo desarrollado.

Capítulo 6

Conclusiones y futuras líneas de trabajo

6.1. Conclusiones

En este trabajo el principal objetivo ha sido solventar la problemática relacionada con la movilidad IPv6. Para ello se ha implementado un *software* que cumple los requisitos del protocolo NEMO BS, especificados en la RFC 3963.

Para poder llevar a cabo dicha implementación, anteriormente ha sido necesario realizar un enorme trabajo en relación a la configuración y despliegue de redes IPv6.

Previo a la implementación de NEMO BS, fue necesario la configuración de los distintos dispositivos que formaban la red. Por ejemplo, a los equipos o PCs se les añadieron distintos paquetes y herramientas para una correcta implementación y evaluación del prototipo. Con esto se adquirieron unos conocimientos en administración de Linux y gestión de redes, de los que antes se carecía. Asimismo, el hecho de trabajar con un tipo de routers diferentes a la anterior implementación (Fonera), ha permitido disponer de WiFi con la distribución *Kamikaze* para un *kernel* 2.6, el cual posibilitaba el soporte para túneles IPv6 sobre IPv6.

También, antes de poder realizar cualquier implementación, se tuvo que investigar sobre la arquitectura de los routers y sobre el nuevo *firmware* OpenWrt, versión *Kamikaze* y sus características, ya que previamente no se había trabajado con él. Además, posibles errores en el cambio del *firmware* podrían provocar un fallo fatal en los routers dejándolos inservibles.

Por otro lado, posteriormente se tuvo que recompilar el *kernel* del agente local añadiendo el paquete de soporte de movilidad de IPv6 aún en fase de experimentación, consiguiendo de esta forma que el HA dejase de generar paquetes ICMP *Parameter Problem* al recibir los mensajes Binding Update procedentes de los routers móviles.

Las pruebas de evaluación han permitido comprobar el correcto funcionamiento del *software*. Por otro lado, los resultados del rendimiento muestran cierta ineficiencia en el tiempo de *handover*, todo apunta a que es causa del *driver* de la tarjeta inalámbrica del router Fonera, debido al tiempo de escaneo cuando se produce un cambio de red, ya que otros tiempos como son el de señalización o el de descubrimiento son mucho menores comparativamente.

El resultado de las pruebas de rendimiento demuestra que el prototipo implementado no influye en la tasa obtenida, tan sólo la parte proporcional de datos de control que se introducen al encapsular la información, desde/hacia la red móvil, a través del túnel. Por tanto se puede concluir que dicho prototipo es eficiente,

Igualmente, esta evaluación experimental del prototipo pone de manifiesto las carencias de la solución NEMO, en cuanto a lo subóptimo de las rutas que éste emplea. Los efectos del *routing* triangular se hacen presentes, más aún cuando la red hogar está lejos de la red visitada. Por la misma razón, un cliente de una red móvil anidada a otra red móvil sufre estos efectos por duplicado. Todo ello hace necesario la implementación de alguna solución que evite el *routing* triangular: soluciones de optimización de rutas como MIRON[1] [BBC04].

Cabe mencionar que el escenario de pruebas configurado se ha hecho de acuerdo al que se implementa dentro del proyecto español POSEIDON [pos] en el cual colabora el Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid. La finalidad de este proyecto es la creación de soluciones de ingeniería para la provisión de servicios sobre redes heterogéneas destinados a usuarios conectados a redes móviles, muy particularmente redes vehiculares, quedando demostrada de forma anterior la compatibilidad del escenario montado en una prueba del proyecto POSEIDON realizada en Barcelona en septiembre de 2009.

Entre las principales dificultades se encuentra la instalación del *firmware* OpenWrt en los routers Fonera, ya que la documentación encontrada ha sido bastante diversa. Además dicha instalación dependía del *firmware* original de los routers.

Otro problema encontrado fue el estudio y entendimiento de la implementación anterior que había sido facilitada, ya que en primer lugar la aplicación no podía ejecutarse debido a fallos de segmentación. En segundo lugar, no era compatible con la solución ideada para el agente local basada en *raw sockets*. A colación de esto, otra dificultad fue el estudio y empleo de los *raw sockets* y sus distintas opciones para el envío y recepción de los mensajes de señalización, como se ha comentado en el Capítulo 3.

Por otro lado, tampoco fue facilitado el *software* que implementaba el protocolo NEMO BS para el agente local, por lo que también se tuvo que proceder a realizar un estudio de sus funciones para implementar una solución compatible con la del router móvil.

La búsqueda de distintos clientes de vídeo fue un trabajo tedioso, ya que en un primer momento se pretendía conseguir *streaming* de vídeo a través de RTSP. Sin embargo, la mayoría de ellos no soportaban este protocolo para IPv6. Y los que lo soportaban, como es el caso del *Real Player 10* para Windows, tenían problemas con el *firewall* o de conectividad. Por estas razones, finalmente en este proyecto se ha decidido utilizar como protocolo de transporte UDP y HTTP para su visualización en navegadores.

A pesar de todo, se puede concluir que la mayoría de los objetivos que se marcaron se han cumplido con creces abriéndose nuevas líneas de investigación como las que se muestran a continuación.

[1] MIRON: *Mobile IPv6 Route Optimisation for Network Mobility*

6.2. Futuras líneas de trabajo

- El soporte de varias interfaces *egress* en el router móvil. Con esto se pretende permitir que el dispositivo se conecte a varias redes, incluso con distintas tecnologías, y así tener más flexibilidad.
- Como extensión de la línea de trabajo anterior probar la implementación en otros tipos de routers que permitan aumentar las funcionalidades ya que el router Fonera tan sólo tiene una interfaz cableada y otra inalámbrica
- A colación del anterior punto, la implementación de optimización de rutas para NEMO diseñada para entornos vehiculares, como es el caso de:
  - VARON. Se realiza optimización de rutas. Para evitar tener que dirigir el tráfico a través de la red hogar, hay mecanismos para detectar cuando un nodo está en las proximidades y poder comunicarse con él por otro interfaz mediante, por ejemplo, una red ad-hoc.
  - MIRON, basado en el procedimiento de *Return Routability* de MIPv6. Esta solución soporta además redes anidadas sin requerir tunelización adicional, con la consiguiente reducción de información de control (*overhead*) y latencia con respecto a NEMO BS [BBC04].
- Otras mejoras como N-PMIPv6 (*NEMO-enabled PMIPv6*) [SBC+09a] que integra de forma completa redes móviles en entornos PMIPv6 (*Proxy Mobile IPv6*) [GLD+08] - dotación de movilidad a dispositivos IP que no tienen soporte para la misma.
- Mejora del tiempo de *handover* mediante la implementación de diferentes soluciones para el *driver* de la interfaz inalámbrica del router Fonera.
- Implementación de mecanismos de seguridad contemplados en la RFC de NEMO BS.
- Desarrollo de una interfaz gráfica que monitorice el movimiento de varios routers móviles al mismo tiempo.

Apéndice A

Planificación de tareas y presupuesto

A.1. Introducción

En este apéndice se presenta una relación de tareas en las que se ha dividido la realización de este proyecto fin de carrera y un desglose justificado de los costes para llevarlas a cabo.

Finalmente, se presenta un presupuesto total de ejecución del proyecto, incluyendo el coste de cada tarea así como de los distintos materiales utilizados.

A.2. Descomposición en tareas

La realización de este proyecto fin de carrera engloba la ejecución de diversas tareas que han sido clasificadas según sus objetivos. Esta división en tareas se presenta en la Tabla A.1.

A continuación, se describen estas tareas, detallando los objetivos específicos, la relación con otras tareas, la duración y el esfuerzo dedicados a cada una de ellas:

- TAREA A: DOCUMENTACIÓN Y ANÁLISIS DEL ESTADO DEL ARTE.
  - Subtarea A.1: OpenWrt y proceso cambio *firmware*
    - Descripción: en esta primera tarea se realiza un estudio del *firmware* OpenWrt, que se instala en el router Fonera, así como del proceso de cambio del mismo.
    - Objetivos: Instalar en los routers Fonera el *firmware* necesario para llevar a cabo la aplicación.
    - Dependencia (o relación) con otras tareas: esta tarea dará comienzo al inicio del proyecto.
    - Duración: 6 semanas.
    - Recursos: Ingeniero 0.5 hombres/mes.
  - Subtarea A.2: entorno de compilación cruzada y desarrollo
    - Descripción: familiarización con el entorno de compilación requerido para el *firmware* de los routers móviles. Se compila el *firmware*, con lo que se

obtiene una imagen del mismo. Con ello se permite compilar aplicaciones para este tipo de routers y además se obtienen los módulos de movilidad necesarios que se han de instalar en dichos routers.

- Objetivos: Compilar el *firmware* de OpenWrt para obtener una imagen del mismo, y poder así desarrollar y compilar aplicaciones para los routers Fonera.
- Dependencia (o relación) con otras tareas: esta tarea dará comienzo tras la tarea A.1.
- Duración: 3 semanas.
- Recursos: Ingeniero 0.5 hombres/mes.

- Subtarea A.3: Protocolos MIPv6 y NEMO
  - Descripción: en esta tarea se ha realizado un estudio previo de la problemática de la movilidad IPv4 e IPv6. Así como de las posibles soluciones o mejoras, siendo en este caso el protocolo NEMO.
  - Objetivos: Con este estudio se pretende tener una visión global de la problemática y posibles soluciones para la mejora en la gestión de la movilidad de redes IP.
  - Dependencia (o relación) con otras tareas: esta tarea dará comienzo tras la tarea A.2.
  - Duración: 2 semanas.
  - Recursos: Ingeniero 0.5 hombres/mes.

- TAREA B: INSTALACIÓN Y CONFIGURACIÓN DE DISPOSITIVOS.
  - Descripción: A lo largo de esta tarea se van a realizar las tareas de cambio del *firmware* a los routers Fonera. También se van a crear los ficheros o *scripts* de configuración para las distintas entidades: agente local, routers de acceso y routers móviles.
  - Objetivos: en esta fase se pretende conseguir que todos los elementos que forman parte del escenario posean el *software* y configuración requerida para posteriormente probar el correcto funcionamiento del *software*.
  - Dependencia (o relación) con otras tareas: esta tarea comenzará una vez que haya terminado la tarea de documentación A.
  - Duración: 5 semanas.
  - Recursos: Ingeniero 0.5 hombres/mes.

- TAREA C: DESARROLLO DE *SOFTWARE*
  - Subtarea C.1: Raw Sockets
    - Descripción: se analizan las distintas tecnologías de *sockets* para implementar el protocolo NEMO BS.
    - Objetivos: Tener una visión global de las diferentes tecnologías para poder llevar a cabo la implementación de dicho protocolo a partir de una tecnología determinada para las distintas entidades.
    - Dependencia (o relación) con otras tareas: esta tarea se comenzará una vez que se haya terminado la tarea B.
    - Duración: 2 semanas.
    - Recursos: Ingeniero 1 hombre/mes.

- Subtarea C.2: NEMO HA
  - Descripción: en esta tarea se programa el *software* correspondiente al agente local. En él se implementa el envío del mensaje BA, la gestión de la *Binding Cache* y del túnel o túneles, así como la escucha de los mensajes BU. En un primer momento se analizan las distintas tecnologías de *sockets* para implementar el protocolo NEMO BS. A continuación se implementa el *software* correspondiente al agente local y al router móvil.
  - Objetivos: Implementación del agente local de acuerdo a los requerimientos del protocolo NEMO.
  - Dependencia (o relación) con otras tareas: esta tarea se comenzará una vez que se haya terminado la tarea C.1.
  - Duración: 2 semanas.
  - Recursos: Ingeniero 1 hombre/mes.
- Subtarea C.3: NEMO MR
  - Descripción: A lo largo de esta tarea se implementa el *software* del router móvil. Sus principales funciones son el envío de mensajes BU, la gestión de varios MNPs y del túnel, y la escucha de los mensajes BA.
  - Objetivos: Realizar la implementación del router móvil de acuerdo a los requerimientos del protocolo NEMO.
  - Dependencia (o relación) con otras tareas: esta tarea se comenzará una vez que se haya terminado la tarea C.2.
  - Duración: 3 semanas.
  - Recursos: Ingeniero 1 hombre/mes.

- TAREA D: DESPLIEGUE Y CONFIGURACIÓN ESCENARIO
  - Descripción: A lo largo de esta tarea se realizará el despliegue de los distintos escenarios, que permitirán en una tarea posterior evaluar el prototipo.
  - Objetivos: Los distintos escenarios permitirán realizar diversas pruebas de validación, cuantitativas y cualitativas, las cuales serán un indicador fundamental de la calidad del *software* desarrollado.
  - Dependencia (o relación) con otras tareas: esta tarea se lleva a cabo tras la tarea C.2.
  - Duración: 3 semanas.
  - Recursos: Ingeniero 0.25 hombres/mes.

- TAREA E: FUNCIONAMIENTO DEL SOFTWARE
  - Subtarea E.1: Funcionamiento
    - Descripción: en esta primera batería de pruebas, se va a certificar el correcto funcionamiento de la aplicación. Por un lado la adecuada señalización y por otro el establecimiento del túnel. Además se procede también a descargar y compilar el kernel de Linux para quitar el mensaje de error (*ICMP Parameter Problem*) comentado en la sección 3.4.
    - Objetivos: Certificar que el *software* desarrollado cumple las especificaciones básicas del protocolo NEMO.
    - Dependencia (o relación) con otras tareas: esta tarea se realizará una vez que termina la tarea D.

- Duración: 3 semanas.
- Recursos: Ingeniero 1 hombre/mes.

- Subtarea E.2: pruebas de estabilidad
  - Descripción: en estas pruebas se pretende comprobar que la aplicación desarrollada funciona de forma estable.
  - Objetivos: Certificar que el *software* no se ve interrumpido por errores de la propia aplicación.
  - Dependencia (o relación) con otras tareas: esta tarea se realizará una vez que termina la tarea E.1.
  - Duración: 2 semanas.
  - Recursos: Ingeniero 1 hombre/mes.
- Subtarea E.3: Soporte de distintos supuestos protocolo (en casa, fuera, cambiar red, varios MNPs, ...)
  - Descripción: A lo largo de estas pruebas se va a comprobar cómo soporta la aplicación otras funciones adicionales definidas en el protocolo.
  - Objetivos: Verificar el funcionamiento de nuevos supuestos, tales como: que el HA soporte varios routers móviles, que el MR soporte varios MNPs, ...
  - Dependencia (o relación) con otras tareas: esta tarea se realiza tras la tarea E.2.
  - Duración: 2 semanas.
  - Recursos: Ingeniero 1 hombre/mes.

- TAREA F: PRUEBAS

  - Subtarea F.1: Autonomía de los routers móviles
    - Descripción: esta tarea se centra en el estudio de la autonomía de los routers Fonera que actúan como MR utilizando dos tipos de alimentación: pilas AA y una batería de 6 V.
    - Objetivos: Medir el tiempo de duración de la autonomía del router móvil para ambos tipos de fuentes de alimentación y distintos casos de uso.
    - Dependencia (o relación) con otras tareas: esta tarea se lleva a cabo tras la tarea E.3.
    - Duración: 2 semanas.
    - Recursos: Ingeniero 0.25 hombres/mes.
  - Subtarea F.2: Rendimiento
    - Descripción: esta tarea evalúa la eficiencia del prototipo para el envío de tráfico entre el router móvil y el agente local. Se prueban distintos tipos de tráfico y diferentes escenarios.
    - Objetivos: Analizar la eficiencia de la implementación realizada.
    - Dependencia (o relación) con otras tareas: esta tarea se lleva a cabo tras la tarea F.1.
    - Duración: 3 semanas.
    - Recursos: Ingeniero 0.5 hombres/mes.
  - Subtarea F.3: *Streaming* de vídeo

- Descripción: esta tarea realiza pruebas de *streaming* de vídeo una vez montado ya el escenario completo (Figura 5.3). Se evalúan distintos clientes de vídeo y se ejecutan unas pruebas cualitativas referentes a la transmisión de vídeo y al tiempo de traspaso entre puntos de acceso.
- Objetivos: evaluación y análisis de la calidad de vídeo transmitido dependiente del tiempo de traspaso.
- Dependencia (o relación) con otras tareas: esta tarea se lleva a cabo tras la tarea F.2.
- Duración: 1 semana.
- Recursos: Ingeniero 1 hombre/mes.

■ TAREA G: TRASPASO O *HANDOVER*

• Subtarea G.1: Simulación cambio de red
  - Descripción: en esta tarea se realizan diferentes simulaciones de cambio de punto de acceso por parte del router móvil, anotando los distintos valores obtenidos para el tiempo de traspaso, que es una medida del rendimiento de la aplicación.
  - Objetivos: Medición del tiempo empleado al cambiar de punto de acceso.
  - Dependencia (o relación) con otras tareas: esta tarea se lleva a cabo tras la tarea F.3.
  - Duración: 1 semana.
  - Recursos: Ingeniero 1 hombre/mes.
• Subtarea G.2: Medidas del tiempo de *handover*
  - Descripción: en esta tarea se calcula dicho tiempo de traspaso (media y desviación típica) a partir de las medidas obtenidas en la tarea G.1. También se analizan y evalúan los resultados obtenidos.
  - Objetivos: cálculo del tiempo empleado al producirse un cambio de punto de acceso.
  - Dependencia (o relación) con otras tareas: esta tarea comienza tras la tarea G.1.
  - Duración: 1 semana.
  - Recursos: Ingeniero 0.5 hombres/mes.
• Subtarea G.3: estudio para la mejora del tiempo de *handover*
  - Descripción: Se realiza el estudio de posibles alternativas para reducir el tiempo de traspaso al producirse un cambio de red.
  - Objetivos: evaluar posibles alternativas que permitan reducir el tiempo de traspaso.
  - Dependencia (o relación) con otras tareas: esta tarea se realiza tras la tarea G.2.
  - Duración: 0.5 semanas.
  - Recursos: Ingeniero 1 hombre/mes.

- TAREA H: MEMORIA
  - H.1 Documentación sobre el desarrollo del proyecto
    - Descripción: en esta tarea se documentan los distintos puntos del proyecto,
    - Objetivos: Documentación de este proyecto.
    - Dependencia (o relación) con otras tareas: esta tarea se realiza tras la tarea G.3.
    - Duración: 2 semanas.
    - Recursos: Ingeniero 1 hombre/mes.
  - H.2 Organización y estructura del documento
    - Descripción: A lo largo de esta tarea se organiza el documento, estructurándolo adecuadamente.
    - Objetivos: Organización y estructura de la memoria del documento.
    - Dependencia (o relación) con otras tareas: esta tarea comienza tras la tarea H.1.
    - Duración: 1 semana.
    - Recursos: Ingeniero 0.5 hombres/mes.
  - H.3 Realización del documento final
    - Descripción: A lo largo de esta tarea se redacta el documento final del proyecto.
    - Objetivos: Redacción de cada uno de los capítulos (y apéndices) de este proyecto.
    - Dependencia (o relación) con otras tareas: esta tarea se lleva a cabo tras la tarea H.2.
    - Duración: 4 semanas.
    - Recursos: Ingeniero 0.5 hombres/mes.
- TAREA I: PRESENTACIÓN
  - Descripción: en esta tarea se va a preparar la presentación de este proyecto.
  - Objetivos: elaborar un conjunto de transparencias adecuado que permitan tener una visión general, clara y completa del trabajado realizado.
  - Dependencia (o relación) con otras tareas: esta tarea se ejecuta tras la tarea H.3.
  - Duración: 2 semanas.
  - Recursos: Ingeniero 0.5 hombres/mes.

Tarea	Duración(s)	Recursos (Ing/m)	Total(h)
Documentación y análisis del estado del arte			
A.1 OpenWrt y proceso cambio *firmware*	6	0.5	120
A.2 Entorno de compilación cruzada y desarrollo *software*	3	0.5	60
A.3 Protocolos MIPv6 y NEMO	2	0.5	40
Total			**220**
Instalación y configuración de dispositivos			
B.1 Routers móviles	1.5	0.5	20
B.2 Agente local	1	0.5	20
B.3 Routers de acceso	1	0.5	20
B.4 Nodo corresponsal	0.5	0.5	10
B.5 Cámara IP(v6)	1	0.5	20
Total			**90**
Desarrollo de software			
C.1 Raw Sockets	2	1	60
C.2 NEMO HA	2	1	80
C.3 NEMO MR	3	1	100
Total			**240**
Despliegue y configuración escenario			
D.1 Despliegue y configuración	3	0.25	30
Total			**30**
Testeo de software			
E.1 Funcionamiento	3	1	120
E.1.1 Sistema operativo Linux y compilación de kernel	1	0.5	20
E.2 Estabilidad	2	1	80
E.3 Soporte distintos supuestos del protocolo	2	1	80
Total			**280**
Pruebas			
F.1 Autonomía	3	0.25	30
F.2 Rendimiento	3	0.5	60
F.3 *Streaming* de vídeo	1	1	40
Total			**220**
Handover			
G.1 Simulación del cambio de red	1.5	1	60
G.2 Medidas del tiempo de *handover*	1	0.5	20
G.3 Estudio para la mejora del tiempo de *handover*	0.5	1	20
Total			**220**
Memoria			
H.1 Documentación sobre el desarrollo del proyecto	2	1	80
H.2 Organización y estructura del documento	1	0.5	20
H.3 Realización del documento final	4	0.5	80
Total			**180**
Presentación			
I.1 Realización de la presentación final	2	0.5	40
Total			**40**
Total			**1470**

Tabla A.1: Resumen descomposición en tareas

A.3. Planificación con el diagrama de fases de ejecución detallado

En la Figura A.1, se presenta el diagrama de *Gantt* reducido de las principales tareas. A continuación en la Figura A.2 se muestra la planificación detallada, según las fases, del proyecto.

Figura A.1: Diagrama de Gantt reducido.

	Nombre	Duración	Inicio	Fin	Predecesoras
1	**Tarea A: DOCUMENTACION**	**55d**	**07/01/2009**	**24/03/2009**	
2	A.1 OpenWrt y procesos de cambio de firmware	30d	07/01/2009	17/02/2009	
3	A.2 Entorno de compilacion cruzada y desarrollo software	15d	18/02/2009	10/03/2009	2
4	A.3 Protocolos MIPv6 y NEMO	10d	11/03/2009	24/03/2009	3
5	**Tarea B: INSTALACION Y CONFIGURACION DISPOSITIVOS**	**25d**	**25/03/2009**	**28/04/2009**	**1,4**
6	B.1 Routers moviles	7.5d	25/03/2009	03/04/2009	4
7	B.2 Agente local	5d	03/04/2009	10/04/2009	6
8	B.3 Routers de acceso	5d	10/04/2009	17/04/2009	7
9	B.4 Nodo correspondiente	2.5d	17/04/2009	21/04/2009	8
10	B.5 Camaras IP(v6)	5d	22/04/2009	28/04/2009	9
11	**Tarea C: DESARROLLO DE SOFTWARE**	**35d**	**29/04/2009**	**16/06/2009**	**5,10**
12	C.1 Raw Sockets	10d	29/04/2009	12/05/2009	10
13	C.2 NEMO HA	10d	13/05/2009	26/05/2009	12
14	C.3 NEMO MR	15d	27/05/2009	16/06/2009	13
15	**Tarea D: DESPLIEGUE Y CONFIGURACION ESCENARIO**	**15d**	**17/06/2009**	**07/07/2009**	**11FI+20d**
16	D.1 Despliegue	15d	17/06/2009	07/07/2009	14
17	**Tarea E: TESTEO DE SOFTWARE**	**35d**	**08/07/2009**	**25/08/2009**	**15**
18	E.1 Autonomia	15d	08/07/2009	28/07/2009	16
19	E.2 Estabilidad	10d	29/07/2009	11/08/2009	18
20	E.3 Soporte distintos supuestos de protocolo	10d	12/08/2009	25/08/2009	19
21	**Tarea F: PRUEBAS**	**30d**	**26/08/2009**	**06/10/2009**	**17**
22	F.1 Autonomia	10d	26/08/2009	08/09/2009	20
23	F.2 Rendiemto	15d	09/09/2009	29/09/2009	22
24	F.3 Streaming de video	5d	30/09/2009	06/10/2009	23
25	**Tarea G: HANDOVER**	**12.5d**	**07/10/2009**	**23/10/2009**	**21**
26	G.1 Simulacion cambio de red	5d	07/10/2009	13/10/2009	24
27	G.2 Medidas del tiempo de handover	5d	14/10/2009	20/10/2009	26
28	G.3 Estudio para la mejora de tiempo de handover	2.5d	21/10/2009	23/10/2009	27
29	**Tarea H: MEMORIA**	**35d**	**23/10/2009**	**11/12/2009**	**25**
30	H.1 Documentacion sobre el desarrollo del proyecto	10d	23/10/2009	06/11/2009	28
31	H.2 Organizacion y estructura del documento	5d	06/11/2009	13/11/2009	30
32	H.3 Realizacion del documento final	20d	13/11/2009	11/12/2009	31
33	**Tarea I: PRESENTACION**	**10d**	**11/12/2009**	**25/12/2009**	**29**
34	I.1 Realizacion de la presentacion final	10d	11/12/2009	25/12/2009	32

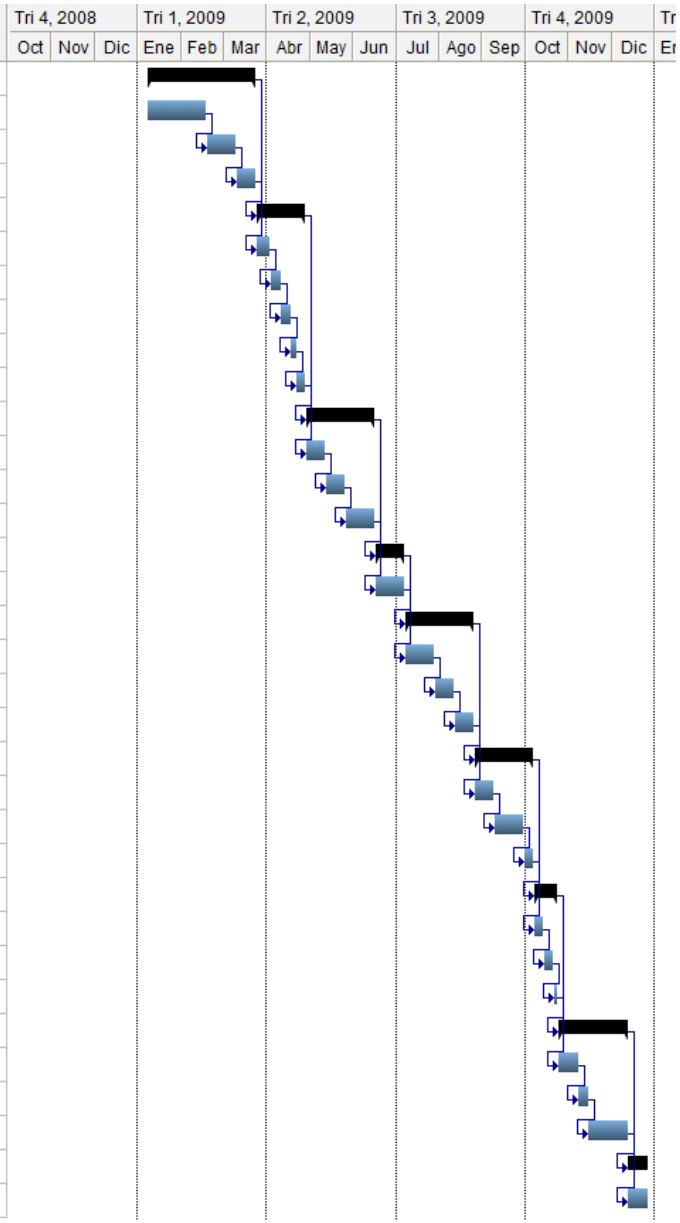

Figura A.2: Diagrama de Gantt.

A.4. Recursos

En esta sección se describen los distintos recursos que son necesarios para la realización del proyecto:

- Recursos materiales:
  - Ordenadores portátiles.
    - Dell Latitute D600. Procesador Intel Pentium 1.73 GHz. 1 GB RAM. Sistema operativo Linux, distribución Ubuntu 8.10.
  - Ordenadores de sobremesa.
    - 2 PCs con procesador Intel Pentium 4 de doble núcleo, 3.4 GHz. 1 GB RAM. Sistema operativo: Linux, distribución Ubuntu 8.04
  - Netbook.
    - Asus Eee PC 1000HE. Procesador Intel Atom N280 a 1,66 GHz. 2 GB RAM. Sistema operativo Linux, distribución Debian 2.6.26
  - Cámara IP: Modelo Axis 207W.
  - Routers Linksys: 2 routers modelo WRT54G.
  - Routers Fonera: 3 routers modelo FON 2200.
  - Pilas: 20 pilas AA alcalinas
  - Batería: De 6V y 4A.
  - Coches RC: 2 coches.
  - Conectores y componentes electrónicos.
  - Equipamiento de red: 1 hub, diversos cables Ethernet.
- Recursos de trabajo: 1 Ingeniero de telecomunicaciones

A.5. Presupuesto de Proyecto

En esta sección se muestra el presupuesto final del Proyecto (Tabla A.2).

1. - Autor: José Pablo Salvador García
2. - Departamento: Ingeniería Telemática
3. - Descripción del Proyecto:
   - Título: implementación de mecanismos avanzados de movilidad sobre routers Fonera
   - Duración: 10 meses
   - Tasa de costes indirectos: no se especifican.
4. - Presupuesto total del Proyecto (valorado en Euros): 44100 euros

Concepto	**Cantidad**	**Coste (€)**	**Total (€)**
Recursos materiales			
Ordenadores portátiles	1	600	600
Ordenadores de sobremesa	2	550	1100
Netbook	1	220	220
Cámara IP	1	320.85	320.85
Routers Linksys	2	60	120
Routers Fonera	3	40	120
Pilas	20	3(x4)	15
Baterías	1	10	10
Coches RC	2	80	160
Conectores y componentes electrónicos	-	10	10
Equipamiento de red (hub, cable Ethernet)	-	-	-
Total			**2675.85**
Recursos de trabajo			
Ingeniero de telecomunicaciones	1 (1470 horas)	30€/hora	44100
Total			**44100**
Otros			
Documentación	-	-	-
Reuniones	-	-	-
Total			-
Total			**46775.85 €**

Tabla A.2: Tabla presupuesto

5. Subcontratación de tareas: no se especifican.
6. Otros costes directos del proyecto: no se especifican.

Apéndice B

Mensajes del protocolo de Soporte Básico de Movilidad de Redes - NEMO BS

En este primer apéndice se va a realizar una descripción más detallada de los mensajes y cabeceras del protocolo de Soporte Básico de Movilidad de Redes, NEMO BS, el cual extiende de MIPv6 donde éstos se encuentran definidos y a los cuales NEMO añade algunos campos adicionales.

B.1. Cabecera de movilidad

La cabecera de movilidad es una cabecera de extensión usada por nodos móviles, nodos corresponsales (CN), en este caso sólo con MIPv6, y agentes locales (HA), en todos los mensajes referentes a la creación y gestión de las asociaciones de movilidad. En la Figura *B*.1 se muestra la estructura de la cabecera de movilidad. Dicha cabecera se indica dentro del campo *Next Header* de la cabecera precedente por un valor de 135.

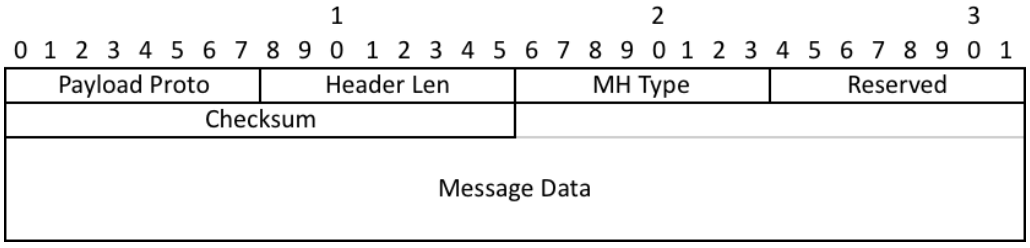

Figura B.1: Formato de la cabecera de movilidad.

Los distintos campos que forman la cabecera de movilidad son:

Payload Proto . Campo de 8 bits que identifica el tipo de cabecera posterior a la cabecera de movilidad. Sus valores son los mismos que los utilizados en el campo *Next Header* de la cabecera IPv6.

Header Len . Entero sin signo de 8 bits, indicando la longitud de la cabecera de movilidad

en unidades de 8 octetos, **excluyendo los primeros 8 octetos**. Esta longitud debe ser un múltiplo de unidades de 8 octetos.

MH Type . Campo de 8 bits, que identifica el tipo de mensaje de movilidad. Los posibles valores de este campo son:

0 *Binding Refresh Request*

1 *Home Test Init*

2 *Care-of Test Init*

3 *Home Test*

4 *Care-of Test*

5 *Binding Update*

6 *Binding Acknowledgement*

7 *Binding Error*

Reserved . Campo de 8 bits reservado para un uso futuro. Su valor debe ser inicializado a cero por el emisor, y debe ser ignorado por el receptor.

Checksum . Entero sin signo de 16 bits, que contiene el resultado de la suma de comprobación de la cabecera de movilidad (*checksum*). Esta suma se calcula con una *pseudo-cabecera*[1] IPv6 seguida de la cabecera de movilidad completa con el campo *checksum* a cero y haciendo el complemento a uno de la suma del complemento a uno de la pseudo-cabecera.

Message Data . Campo de longitud variable que contiene los datos específicos del mensaje de movilidad.

B.2. Binding Update

Este tipo de mensaje es enviado por el router móvil (*Mobile Router, MR*) al agente local (*Home Agent, HA*) para indicarle su punto de acceso tan pronto como el MR haya adquirido su *Care-of Address* o CoA al recibir el *router advertisement* procedente del router de acceso (*Access Router*) de la red visitada.

La Figura B.2 muestra la estructura de un mensaje *Binding Update*.

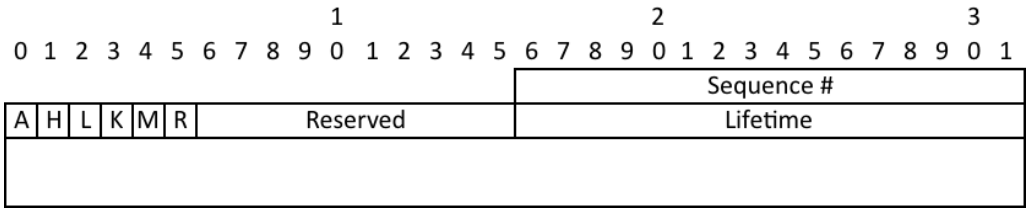

Figura B.2: Formato de mensaje Binding Update.

Sequence . Número de secuencia de 16 bits. Debe ser replicado en los *Binding ACK* (BA) de respuesta.

[1]La pseudo-cabecera contiene campos de la cabecera IPv6: el campo *Next Header* debe tener valor dos; las direcciones deben ser las que aparecen en los campos dirección fuente y dirección destino del paquete IPv6 que contiene la cabecera de movilidad

A (*Acknowledge*). Flag que solicita el envío de un BA como respuesta.

H (*Home Registration*). Indica al receptor la petición del MR de que actúe como su HA, agente local.

L (*Link-Local Address Compatibility*). Este flag es establecido cuando la *Home Address* o dirección en la red hogar indicada por el nodo móvil tiene el mismo identificador de interfaz que la dirección local del router de acceso.

K (*Key Management Mobility Capability*). Indica si la asociación IPsec utilizada en el túnel debe eliminarse y volverse a crear cuando el nodo se mueve, o persiste a estos movimientos. Un valor cero en este bit indica que la asociación IPsec no sigue en uso en los movimientos de red.

M (*Mobility Session*). Este flag está reservado para el protocolo HMIPv6[2].

R (*Mobile Router Flag*). Este flag se incluye para indicar al agente local si el *Binding Update* proviene de un nodo o un router móvil. Si tiene valor cero el agente local supondrá que el router móvil está funcionando como nodo y no le reenviará los paquetes destinados a la red móvil.

Reserved . 10 bits que no se utilizan. Deben ser inicializados a cero e ignorados en recepción.

Lifetime . Entero sin signo de 16 bits que indica el tiempo de vida de la asociación. Se mide en unidades de 4 segundos. Un valor cero en este campo indica que la entrada en la *Binding Cache* tiene que ser borrada (el router móvil se encuentra en la red hogar - proceso de desasociación).

Mobility Options . Campo de longitud variable que contiene cero o más opciones en formato TLV (tipo-longitud-valor, *Type-Length-Value*). Las opciones válidas de un *Binding Update* son:

- *Nonce Indices Option.*
- *Alternate Care-of-Address Option.*
- *Unique Identifier Option.*
- *Binding Authorization Data Option*: Obligatoria en los *Binding Update* enviados a un nodo corresponsal.

Si no hay ninguna opción se deben incluir cuatro octetos de *padding* o relleno y es necesario poner el campo *Header Len* de la cabecera de movilidad a 1. Para más información ver la sección B.5.

B.3. Binding ACK

Este mensaje se utiliza como asentimiento de un *Binding Update*. La Figura *B.3* muestra la estructura de un mensaje *Binding Acknowledgement*.

[2]HMIPv6: ***H****ierarchical* ***M****obile* ***IPv6***

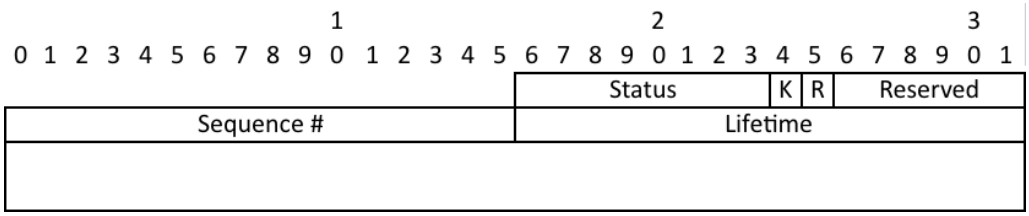

Figura B.3: Formato de mensaje Binding ACK.

Status . Entero sin signo de 8 bits que indica el estado de la petición del *Binding Update*. Valores menores de 128 indican que dicho *Binding Update* fue aceptado y mayores que 128 que fue rechazado. A continuación se muestran con mayor detalle los posibles estados de este campo:

0 *Binding Update accepted*

1 *Accepted but prefix discovery necessary*

128 *Reason unspecified*

129 *Administratively prohibited*

130 *Insufficient resources*

131 *Home registration not supported*

132 *Not home subnet*

133 *Not home agent for this mobile node*

134 *Duplicate Address Detection failed*

135 *Sequence number out of window*

136 *Expired home nonce index*

137 *Expired care-of nonce index*

138 *Expired nonces*

139 *Registration type change disallowed*

140 *Mobile Router Operation not permitted*

141 *Invalid Prefix*

142 *Not Authorized for Prefix*

143 *Forwarding Setup failed (prefixes missing)*

R (*Mobile Router Flag*). Indica que el agente local que procesó el mensaje *Binding Update* soporta routers móviles. Este flag se establece a uno si el correspondiente BU también lo tenía a uno.

Sequence . El número de secuencia debe coincidir con el del correspondiente *Binding Update*, para que el router móvil sepa a qué petición pertenece.

Lifetime . Debe ser el mismo que el del correspondiente *Binding Update*.

K* y *Reserved . Igual que en el *Binding Update*.

B.4. Binding Refresh

Este mensaje es generado por el router móvil cada cierto tiempo. Se trata simplemente de un *Binding Update* que es recibido por el agente local, actualizando el valor de la entrada correspondiente a ese router móvil en su *Binding Cache* o tabla de asociaciones.

La Figura B.2 muestra la estructura de este tipo de mensajes.

B.5. Opciones de movilidad

Las opciones de movilidad permiten incluir información adicional que puede no ser necesaria en todos los mensajes, así como permitir futuras extensiones en el formato. Su uso no es obligatorio y también se permite añadir varias de estas opciones incluidas dentro del campo de datos del mensaje, indicándolo en el campo *Header Len* de la cabecera de movilidad. Cuando el valor de este campo es mayor que la longitud requerida para el tipo de mensaje de movilidad (definido en el campo *MH Type* de la cabecera de movilidad), los restantes octetos se interpretan como opciones de movilidad o *mobility options*. Para más información se puede consultar la sección 6.2 de [JPA04].

Estas opciones son codificadas usando triplas TLV (*Type, Length, Value*). En la Figura B.4 se muestra el formato de este tipo de mensajes.

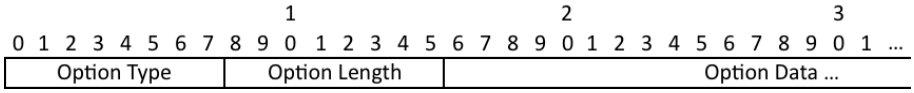

Figura B.4: Formato TLV para opciones de movilidad.

1. *Option Type*: Identificador de 8 bits del tipo de opción de movilidad.
2. *Option Length*: entero sin signo de 8 bits, que indica la longitud de la opción en octetos, sin contar el campo *Type* ni el campo *Length*.
3. *Option Data*: Campo de longitud variable que contiene la información correspondiente a dicha opción.

A continuación se especifican las opciones que están definidas actualmente para ser incluidas en los mensajes de movilidad. Las opciones de movilidad pueden requerir alineamiento, de forma que campos de anchura n octetos se coloquen en un número entero múltiplo de n, a contar desde el principio de la cabecera, para n = 1, 2, 4, ó 8.

- Pad1: esta opción se utiliza para insertar un octeto de relleno (*padding*). Es una opción especial por lo que no tiene campo *Option Length* ni *Option Data*, además de no requerir alineamiento. El formato de esta opción se muestra en la Figura B.5.

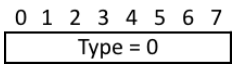

Figura B.5: Formato de la opción de movilidad Pad1.

- PadN: Se utiliza para insertar dos o más octetos de relleno. El campo *Option Length* contiene el valor de N-2, siendo N el número de octetos que se quieren añadir. El campo *Option Data* está formado por N-2 octetos con valor cero. Esta opción tampoco requiere alineamiento. Su formato se observa en la Figura B.6.

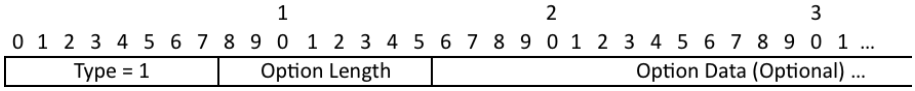

Figura B.6: Formato de la opción de movilidad PadN.

- *Alternate Care-of Address*. Normalmente, un *Binding Update* especifica la *Care-of Address* en el campo de la dirección origen (*Source Address*) de la cabecera IPv6. Sin embargo, en algunos casos esto no es posible, por ejemplo cuando el nodo móvil pretende indicar una dirección que topológicamente no pueda ser utilizada como dirección origen, o cuando la cabecera no se encuentra protegida por mecanismos de seguridad. En estos casos se puede utilizar esta opción, que contiene una dirección que se usará como *Care-of Address* en el proceso de registro en el agente local, en lugar de utilizar la dirección origen del paquete IPv6. Esta opción debe estar alineada a 8n+6. Su formato se muestra en la Figura B.7.

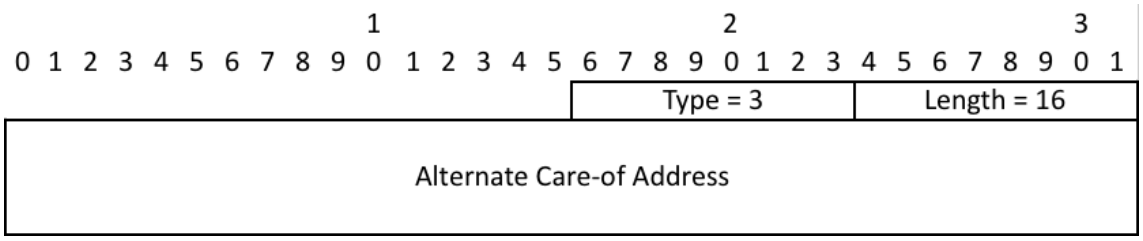

Figura B.7: Formato de la opción de movilidad *Alternate Care-of Address*.

- Otras opciones son: *Binding Refresh Advice, Nonce Indices, Binding Authorization Data*.

B.6. Modos Implícito y Explícito

Cuando el router móvil envía un *Binding Update* al agente local es necesario que se especifiquen los prefijos de la red móvil. Existen dos variantes para realizar este proceso, conocidas como modo explícito e implícito.

En el modo implícito el router móvil no incluye ningún prefijo de red (*Network Prefix*) en el mensaje *Binding Update*. El agente local puede utilizar distintos procedimientos (no definidos por la RFC 3963 [DWPT05]) para determinar los prefijos correspondientes a la red móvil pertenecientes a dicho router móvil y configurar el reenvío de mensajes a esa red. Una forma sería establecer una configuración manual de dichos prefijos en el agente local, como se ha hecho en este caso, mediante ficheros de configuración.

En el modo explícito el router móvil incluye en el *Binding Update* uno o más prefijos correspondientes a la red móvil (*Mobile Network Prefix Options*). Estas opciones incluyen información acerca de los prefijos configurados en dicha red.

B.7. Home Address Option

Esta opción se encuentra dentro de la cabecera de extensión *Destination Option* con el valor del campo *Next Header* de la cabecera IPv6 a 60. Es utilizado por el nodo móvil mientras está fuera de casa, para informar al agente local de su dirección en la red hogar. Esta opción se codifica de acuerdo al formato de una tripla TLV (Figura B.8) con los siguientes valores:

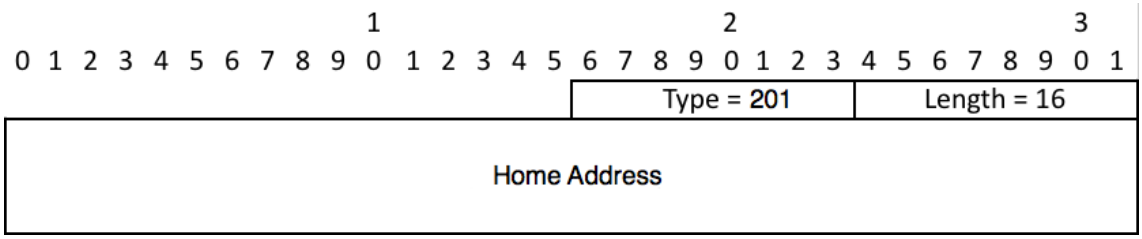

Figura B.8: Formato de la opción *Home Address*.

Option Type . Campo con valor 201 en decimal, o 0xC9 en hexadecimal.

Option Length . Campo con un valor fijado a 16.

Home Address . La dirección en la red hogar del nodo móvil que envía al agente local. Esta dirección debe ser unicast.

El alineamiento requerido para la opción *Home Address* es de 8n+6. Además, dicha opción debe incluirse de acuerdo a algunas restricciones:

- Debe aparecer después de la cabecera *Routing Header* si ésta está presente.
- Debe incluirse antes de la cabecera *Fragment Header* si ésta está presente.
- Debe incluirse antes de la cabecera *AH Header* o *ESP Header* si una de estas dos está presente.

Apéndice C

Instalación de OpenWrt en la Fonera

C.1. El *firmware* OpenWrt

El *firmware* es un bloque de instrucciones de programa para propósitos específicos, grabado en una memoria de tipo no volátil (ROM, EEPROM, *flash*,...), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Funcionalmente, el *firmware* es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica.

El *firmware* se encuentra en memorias ROM de los sistemas de diversos dispositivos periféricos, como en monitores de vídeo, unidades de disco, impresoras, etc., pero también en los propios microprocesadores, chips de memoria principal y en general en cualquier circuito integrado.

En un microprocesador el *firmware* es el que recibe las instrucciones de los programas y las ejecuta en la circuitería del mismo, emitiendo órdenes a otros dispositivos del sistema. Muchos de los *firmwares* almacenados en ROM están protegidos por Derechos de Autor.

OpenWrt (Figura *C*.1) es un *firmware* libre, basado en GNU/Linux optimizado para routers con un *hardware* determinado y reducidas capacidades. El proyecto comenzó en 2004, basándose en un primer momento en el código fuente del router Linksys WRT54G, como mera referencia. Pero pronto se comenzó a dar soporte a modelos parecidos, incluso de otros fabricantes y de ahí a diferentes arquitecturas y *chipsets*.

Básicamente, OpenWrt permite elegir dos tipos de sistemas de archivos:

- Squashfs: Un sistema de ficheros que proporciona una partición de sólo lectura. Esta partición contiene un entorno Linux mínimo, preparado para arrancar el router y proveer lo básico. Esto hace necesaria una partición JFFS2 para poder configurar el router, es decir, para posibilitar volver a la configuración por defecto si por un fallo se deja éste inaccesible.

- JFFS2: Un sistema de ficheros con una partición de lectura y escritura. Especialmente diseñado para transacciones en memorias *flash*. Todo lo que no provee la partición *squashfs* puede ser incorporado en esta partición.

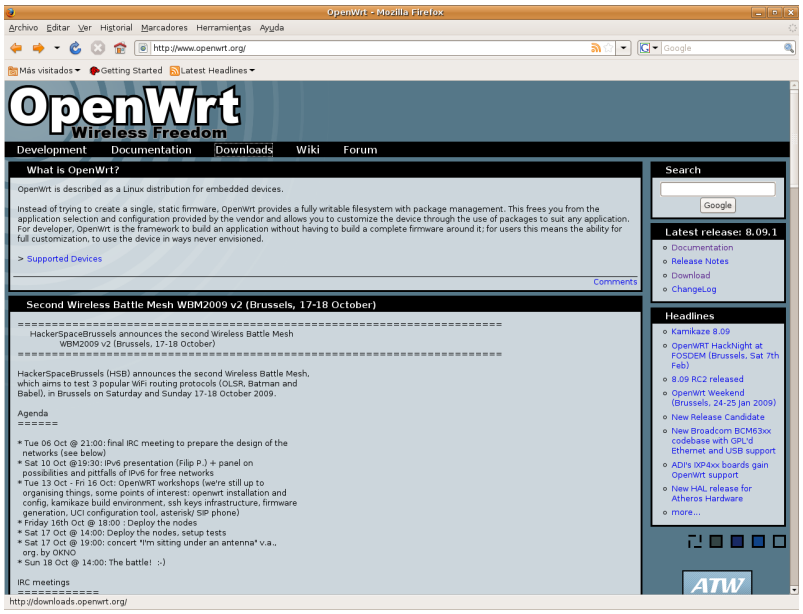

Figura C.1: Página web de OpenWrt.

OpenWrt tiene dos líneas de desarrollo[1]:

- *WhiteRussian*. Distribución ya estable, que utiliza un kernel Linux 2.4.x y sólo soporta unas pocas arquitecturas *hardware*.
- *Kamikaze*. Es la que más se está desarrollando actualmente. Utiliza kernel Linux 2.4 y 2.6. Soporta más arquitecturas, por lo tanto, mayor número de dispositivos. Kamikaze v8.09.1 es la última versión estable con fecha de julio de 2009.

Debido a que la distribución *WhiteRussian* no soporta un kernel Linux 2.6 y tampoco tiene desarrollo en la actualidad (última actualización con fecha de febrero de 2007 versión rc6) se descarta su uso. Por tanto, se utilizará el *firmware Kamikaze* que es el *codename* o nombre de la versión 2.0 del *firmware* OpenWrt. Tiene soporte para el chip *Atheros* que usa la Fonera y es una opción interesante como *firmware* debido a su creciente desarrollo. También es indispensable esta versión de kernel, como ya se comentado en la subsección 3.2.1, para dar soporte para túneles IPv6 sobre IPv6.

C.2. El router Fonera 2200

C.2.1. El proyecto FON

El proyecto FON (Figura C.2) conforma la mayor comunidad *WiFi* del mundo. A través de los routers inalámbricos de FON, conocidos como *Foneras*, se permite a la gente compartir una pequeña parte del ancho de banda de su acceso a Internet con el resto de la comunidad FON. A cambio de esto, los miembros de esta comunidad, conocidos como *Foneros*, obtienen acceso gratuito en los puntos de acceso FON en todo el mundo. FON, entre cuyos inversores se encuentran *Google, Skype* o *British Telecom*, ha crecido de forma

[1] Actualmente, existe una tercera línea de desarrollo *Backfire*.

vertiginosa hasta alcanzar los 800.000 miembros y 300.000 *hotspots* o puntos de acceso desde su fundación en 2006. [ope]

Figura C.2: Logo del proyecto FON.

C.2.2. El router Fonera

La Fonera (router FON 2200) viene actualmente con el *firmware* 2.4.x OpenWrt modificado por defecto. En la Tabla C.1 se muestran las especificaciones del router FON 2200:

Arquitectura	MIPS
Vendor	Atheros
CPU Speed	180 MHz
System-On-Chip	Atheros AR2315
Dimensiones	93.5 mm x 25.5 mm x 70 mm (excluyendo antena)
Alimentación	Entrada: 100-240V 50-60 Hz 0.3A. Salida: 7.5V, 1A DC
Consumo	4 Watios
Memoria	Flash: 8 MB / SDRAM: 16 MB
Conector de antena	Conector RP-SMA (SMA inverso)
Antena	Antena omnidireccional desmontable (2 dBi)
Autenticación	WEP 64bit/128 bit, WPA, WPA2, WPA mixed
Cifrado	TKIP, AES, Mixed
Estándares	IEEE 802.11b / 802.11g (hasta 54 Mbps)
Puertos	LAN/WAN: Un puerto 10/100 RJ-45
USB	No
Serial	Sí
JTAG	Sí

Tabla C.1: Especificaciones técnicas del router Fonera 2200

A continuación se van a explicar en detalle los pasos para cambiar el *firmware* de un router Fonera. Se ha decidido instalar la versión *Kamikaze* 7.09 de OpenWrt. Nótese que se tiene en cuenta que la versión del *firmware* original es 0.7.1rev2, porque para otras versiones cambiaría dicho proceso.

C.3. Proceso de cambio del *firmware*

OpenWrt es una distribución de Linux muy ligera pensada para ser instalada en dispositivos con recursos muy limitados, como por ejemplo routers, con el fin de obtener el

máximo rendimiento y explotar así todo su potencial desarrollar toda su funcionalidad. Por lo tanto, instalar OpenWrt en la Fonera tiene innumerables beneficios. En este apartado se especifican con mayor detalle los pasos a seguir para realizar el cambio del *firmware*.

C.3.1. Configuración interna del router Fonera

En este primer punto se va a proceder a cambiar la dirección IP de la Fonera y su servidor DNS, para evitar que se descarguen nuevas actualizaciones del *firmware* de FON al conectarse a Internet. Para ello se realizan los siguientes pasos:

1. Conectar el router Fonera directamente con un ordenador mediante el cable que se provee en la caja (uso de la interfaz Ethernet), sin tener conexión a Internet (Figura C.3). También se podría hacer a través de la interfaz inalámbrica.

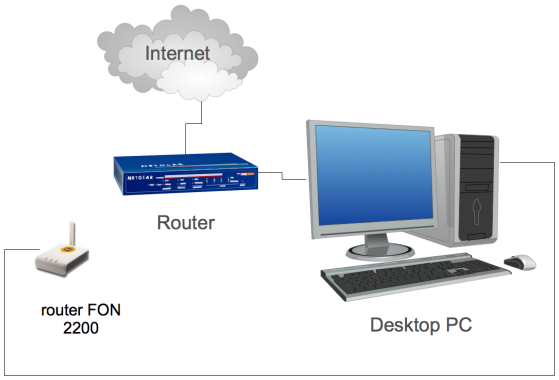

Figura C.3: Escenario del cambio de *firmware* del router Fonera.

2. Configurar la dirección IP del equipo a una del rango 169.254.255.X, ya que la dirección IP para acceder a la Fonera mediante la interfaz Ethernet es 169.254.255.1. En este caso, por ejemplo, la IP elegida es 169.254.255.5.

3. A continuación se accede a la Fonera, para ello introducir en un navegador: http://169.254.255.1. Cuando sean requeridos los datos de usuario y contraseña se deberá introducir:

   ```
   User: admin
   Password: admin
   ```

4. Una vez accedido a la interfaz web de configuración del router (Figura C.4) en el menú de la izquierda seleccionar la opción *Advanced*, y dentro de ésta *Internet Connection*. Se configura el modo IP estático (*Mode Static IP*) y se completan los diferentes campos con los siguientes valores:

   ```
   IP: 192.168.0.2 (Dirección IP en el rango del router que da acceso
   a Internet)
   Mask: 255.255.255.0
   Gateway: 192.168.0.1 (Dirección IP del router)
   DNS Server: 88.198.165.155 (Hack de Kolofonium)[2]
   ```

[2]Esta dirección ha cambiado en posteriores versiones. Para evitar tener que conocer la dirección y posibles cambios en ella, se recomienda resolver la dirección IP correspondiente a kolofonium.datenbruch.de

El *Hack de Kolofonium* es un pequeño servidor DNS que redirige las peticiones al servidor de FON (*radius01.fon.com*) a un servidor *radius* que proporciona al router los datos de configuración necesarios para habilitar el acceso por SSH[3] en lugar de los datos de configuración inicial de FON.

Una vez que se reinicia la Fonera el demonio SSH debería arrancar automáticamente, pudiendo cambiar la configuración DNS de acuerdo a las necesidades.

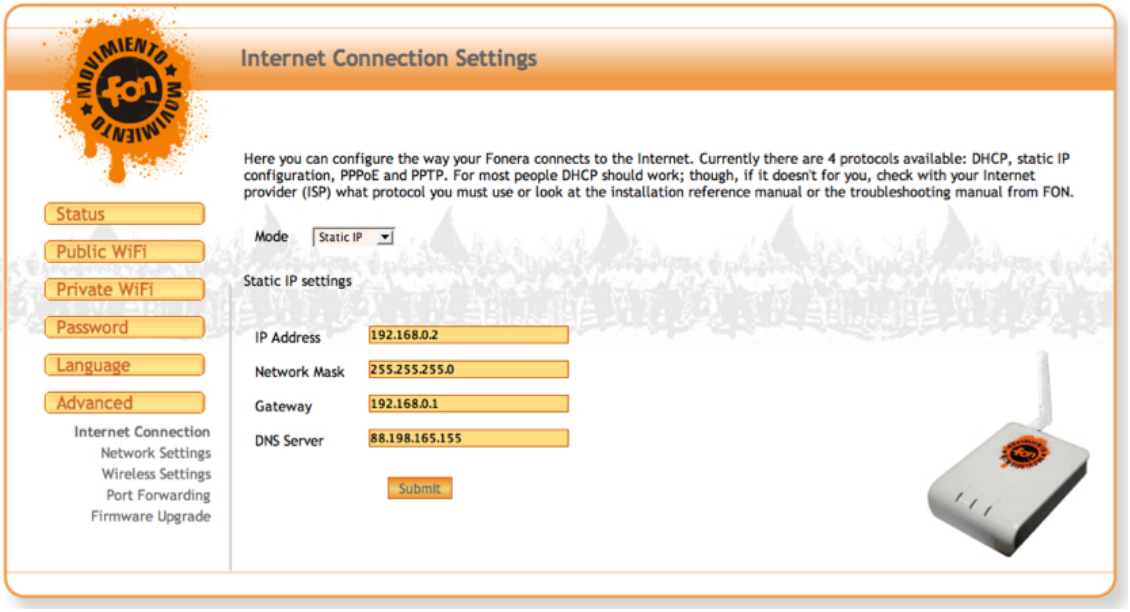

Figura C.4: Página de configuración de la Fonera.

5. Actualizar la configuración pulsando el botón Submit. Nótese que hay que esperar unos minutos a que la Fonera arranque y el DNS de *Kolofonium* haga que se ejecute *dropbear*[4]. Transcurridos unos minutos se puede comprobar que la configuración de la Fonera es la que se observa en la Figura C.5.

C.3.2. Acceso mediante SSH

En este segundo paso se va a habilitar el acceso mediante SSH. También se caparán las actualizaciones del *firmware*.

1. Tras los cambios realizados en la configuración se desconecta el cable de red entre el router Fonera y el PC. Ahora el router Fonera se conecta al PC mediante el router (configurar la IP del ordenador como automática, que se asigne mediante DHCP[5]). Se espera unos minutos hasta que se creen las redes *FON_ AP* y *MyPlace*.

[3] ***S****ecure* ***SH****ell*: Nombre del protocolo y programa que lo implementa. Sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo el ordenador mediante un intérprete de comandos, copiar datos de forma segura, gestionar claves RSA y pasar los datos de cualquier otra aplicación por un canal seguro.

[4] Mini servidor y cliente SSH, diseñado para dispositivos con memoria y recursos limitados, tales como dispositivos embebidos o empotrados.

[5] ***D****ynamic* ***H****ost* ***C****onfiguration* ***P****rotocol*: Protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración automáticamente.

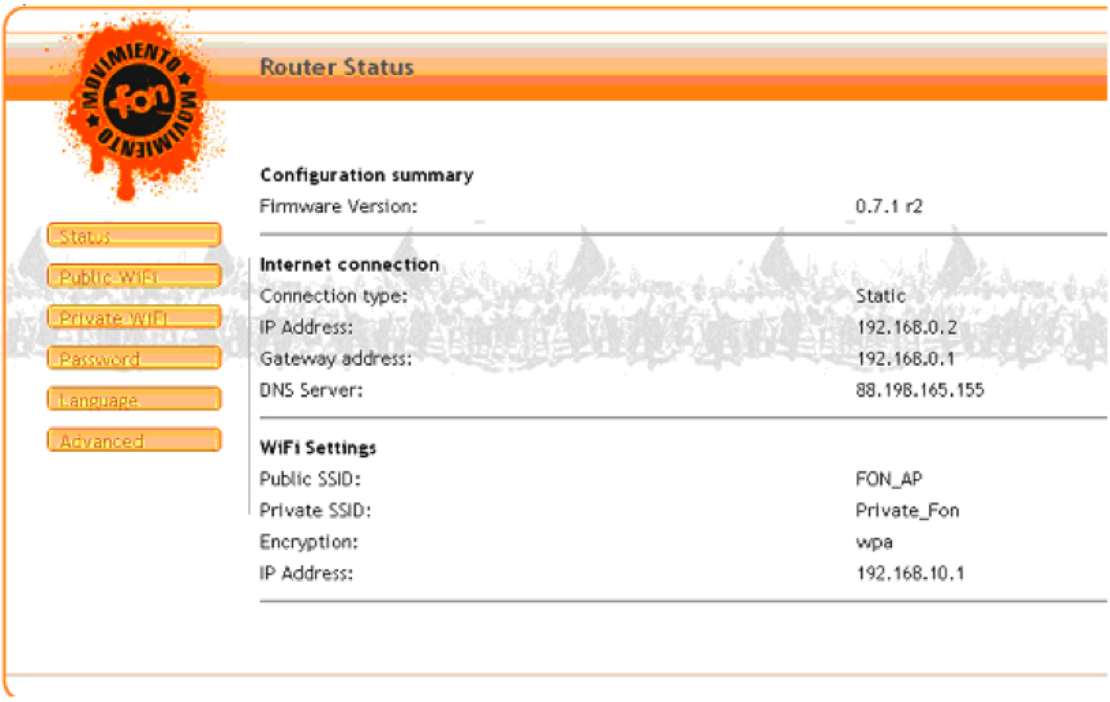

Figura C.5: Resultado de la configuración aplicada.

Conectarse a la red privada de la Fonera identificada por la SSID[6] *MyPlace*. La contraseña es la S/N del router Fonera que se encuentra en su parte inferior o en un lado de la caja.

2. Se abre un terminal y se conecta a la Fonera mediante SSH:

```
ssh root@192.168.10.1 (192.168.10.1: dirección de la interfaz
inalámbrica de la Fonera)
Password: admin
```

En este punto se pueden encontrar problemas si se ha accedido anteriormente a otros routers Fonera. Para ello hay que borrar el fichero de *known_hosts* especificado en el mensaje de error, en el que aparece la ruta o *path* dónde se encuentra (*ficherohosts_ruta*) y se procede a eliminarlo:

```
rm -f ficherohosts_ruta
```

3. Ahora ya se ha accedido al router. Se sube un nivel, situándose en el directorio principal (Figura C.6).

4. Lo siguiente es conseguir que se ejecute *sshd* al inicio, para ello se realiza la siguiente modificación:

```
root@OpenWRT:/# ln -s /etc/init.d/dropbear /etc/init.d/S50dropbear
```

[6] ***S****ervice* ***S****et* ***ID****entifier* es un código incluido en todos los paquetes de una red inalámbrica para identificarlos como parte de esa red. El SSID (Service Set IDentifier) es un código incluido en todos los paquetes de una red inalámbrica (Wi-Fi) para identificarlos como parte de esa red, a menudo conocido como nombre de la red.

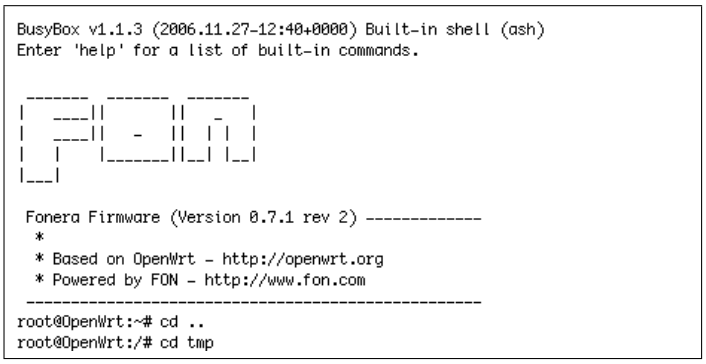

Figura C.6: Vista del directorio raíz del router Fonera.

A continuación se ejecuta el fichero una vez modificado:

```
root@OpenWRT:/# /etc/init.d/S50dropbear
```

Con esto se consigue que *dropbear* se ejecute automáticamente en cada inicio.

5. Ahora se edita el fichero *firewall.user* pulsando `ESC i`:
   `root@OpenWRT:/# vi /etc/firewall.user`

   5.1 Se descomentan las siguientes líneas referentes al *iptables*.

   ```
   # iptables -t nat -A prerouting_rule -i $WAN -p tcp --dport 22 -j ACCEPT
   # iptables -A input_rule -i $WAN -p tcp --dport 22 -j ACCEPT
   ```

   Esto permite que se escuche por el puerto 22 para el acceso SSH.

   5.2 Se añaden las líneas que permiten no tener que conectarse a *MyPlace* para acceder por SSH y poder hacerlo también a través de la interfaz LAN.

   ```
   iptables -t nat -A prerouting_rule -i $LAN -p tcp --dport 22 -j ACCEPT
   iptables -A input_rule -i $LAN -p tcp --dport 22 -j ACCEPT
   ```

   5.3 Una vez terminado, se guardan los cambios (`ESC wq!`) y se ejecuta de nuevo el fichero modificado:

   ```
   root@OpenWRT:/# /etc/firewall.user
   ```

6. Ahora se capan las actualizaciones del *firmware*:

   ```
   root@OpenWRT:/# vi /bin/thinclient
   ```

   Se edita el fichero (`ESC i`) y se comenta la última línea, quedando:

   ```
   # . /tmp/.thinclient.sh
   ```
   Una vez terminado se sale y se guardan los cambios.

7. Por último, se reinicia el router Fonera, para ello se desconecta el cable de la alimentación o se teclea *reboot*.

C.3.3. Instalar *RedBoot* en la Fonera

En este tercer paso se va a instalar *RedBoot*[7] en la Fonera. Es una aplicación de código abierto que permite el intercambio de archivos en sistemas empotrados mediante el cable serie o el cable Ethernet. Está provisto de una línea de comandos que permite administrar las imágenes de la memoria *flash*, las configuraciones del *RedBoot* o descargar archivos desde un servidor TFTP.

1. Se vuelve a acceder a la Fonera mediante SSH, pero ahora se conecta de nuevo la Fonera al PC (dejar que la dirección IP del PC sea asignada mediante DHCP) mediante el router (Figura C.7) para que así la Fonera tenga acceso a Internet:

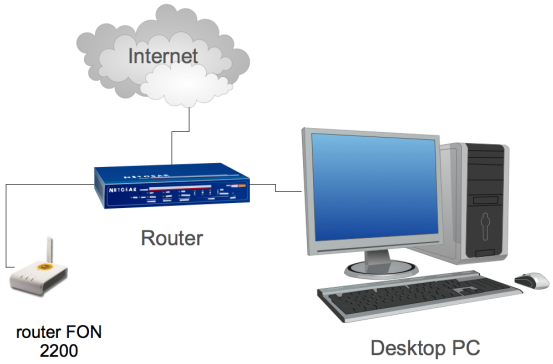

Figura C.7: Escenario del cambio de *firmware* del router Fonera.

```
ssh root@169.254.255.1 (169.254.255.1: Dirección de la interfaz
Ethernet de la Fonera)
password: admin
```

2. Se accede al directorio temporal del router:

```
root@OpenWRT:~#
```

3. Debido a que no se puede escribir en la partición *mtd* correspondiente a la configuración de *RedBoot*, se descarga un kernel de FON modificado para permitir la escritura en la partición de configuración de *RedBoot*, para después modificar esta partición consiguiendo acceder a la consola de *RedBoot*:

```
root@OpenWRT:~# wget http://ipkg.k1k2.de/hack/openwrt-ar531x-2.4-vmlinux-CAMICIA.lzma
```

4. Se instala el nuevo kernel:

```
root@OpenWRT:~# mtd -e vmlinux.bin.l7 write openwrt-ar531x-2.4-vmlinux-CAMICIA.lzma
vmlinux.bin.l7
```

5. Por último se reinicia el router: `root@OpenWRT:~# reboot`

6. Nuevamente se accede a la Fonera mediante SSH, pero ahora se conecta de nuevo la Fonera al PC (Configurar la dirección del PC a 169.254.255.5) mediante el router.

[7] *RedBoot* ***(Red Hat Embedded Debug and Bootstrap firmware)***: Cargador de arranque utilizado en sistemas embebidos. Se tendrá un tiempo limitado en el arranque de la Fonera para tener acceso a él.

Se teclea:

```
ssh root@169.254.255.1 (Dirección de la interfaz Ethernet de la
Fonera)
password: admin
```

7. En este punto se encuentra en del directorio temporal del router Fonera:

   ```
   root@OpenWRT:~#
   ```

8. Se descarga el fichero para habilitar el *RedBoot*:

   ```
   root@OpenWRT:~# wget http://ipkg.k1k2.de/hack/out.hex
   ```

9. Lo siguiente es montar dicho fichero:

   ```
   root@OpenWRT:~# mtd -e 'RedBoot config' write out.hex 'RedBoot config'
   ```

10. Finalmente se reinicia el router: `root@OpenWRT:~ reboot`

    En esta parte es mejor quitar el cable para tener tiempo para acceder al *RedBoot*. Tras el último comando la Fonera se reiniciará pero no podrá completar su arranque por lo que dejará de ser accesible por SSH. No pasa nada, porque durante los 10 primeros segundos de arranque será accesible por *RedBoot*.

C.3.4. *Flashear* y cargar el nuevo *firmware* en la Fonera

El último paso es *flashear* la Fonera y cargar el nuevo *firmware*.

1. Se configura el equipo con la dirección: 192.168.1.X (En mi caso: 192.168.1.5)
2. Se conecta el router Fonera y el ordenador mediante el cable proporcionado en la caja de la Fonera.
3. Se arranca el servidor TFTP que contiene el sistema de ficheros y el nuevo kernel.
4. Entrar en el *RedBoot* de la Fonera: `telnet 192.168.1.254 9000`[8]
5. Cuando conecte, hacer CTRL+C y se entra al *RedBoot*. De esta forma se detiene el arranque y se puede comenzar a *flashear* la Fonera. Si no da tiempo a conectar a *RedBoot*, desenchufar y volver a conectar la alimentación de la Fonera, así se dispondrá de otros 10 segundos. Si todo va bien se verá el prompt RedBoot>".

   ```
   ~ josepablosalvador$ telnet 192.168.1.254 9000
   Trying 192.168.1.254...
   Connected to 192.168.1.254.
   Escape character is '^]'.
   ^C
   ```

   Otra posible salida por pantalla sería:

[8] Puerto en el que escucha *RedBoot*.

```
~ josepablosalvador$ telnet 192.168.1.254 9000
Trying 192.168.1.254...
Connected to 192.168.1.254.
Escape character is '^]'.
== Executing boot script in 9.560 seconds - enter ^C to abort
RedBoot> No image 'vmlinux.bin.l7' found
RedBoot> Can't execute Linux - invalid entry address
```

No preocuparse por esas 2 últimas líneas, el proceso sigue funcionando correctamente.

6. Se configura la IP del servidor TFTP para cargar los ficheros, mediante una IP local (cliente, en este caso, la Fonera) y la IP del servidor TFTP:
ip_addr -h @TFTP server -l @Fonera
ip_addr -h 192.168.1.X(X=5) 192.168.1.254/24

```
RedBoot> ip_addr -h 192.168.1.5 -l 192.168.1.254/24
IP: 192.168.1.254/255.255.255.0, Gateway: 0.0.0.0
Default server: 192.168.1.5
```

Ahora ya se puede *flashear* la Fonera desde *Redboot* con la siguiente secuencia[9]:

7. Se crea una nueva tabla *fis* (*flash image system*), es como si fuera el *Partition Directory*. Esto es necesario, ya que el esquema de la memoria *flash* es diferente en OpenWrt.

```
RedBoot> fis init
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
... Erase from 0xa87e0000-0xa87f0000: .
... Program from 0x80ff0000-0x81000000 at 0xa87e0000: .
```

8. Se transfieren los ficheros al *BootLoader*. Se carga la imagen del sistema de ficheros.

```
RedBoot> load -r -v -b 0x80041000 openwrt-atheros-2.6-root.jffs2-64k
Using default protocol (TFTP)
-
Raw file loaded 0x80041000-0x801e0fff, assumed entry at 0x80041000
```

9. Se crea la imagen *flash* del sistema de ficheros.

```
RedBoot>fis create -b 0x80041000 -f 0xA8030000 -l 0x006F0000 -e 0x00000000
rootfs
... Erase from 0xa8030000-0xa8720000:
.....................................................................................
... Program from 0x80041000-0x80731000 at 0xa8030000:
.....................................................................................
... Erase from 0xa87e0000-0xa87f0000: .
... Program from 0x80ff0000-0x81000000 at 0xa87e0000: .
```

10. Se transfiere el kernel al *BootLoader*.

[9] Si en algún momento algún comando da error de argumento inválido, quitar el cable y volver a conectar procurando que el último comando hecho con éxito sea un *fis*.

```
RedBoot> load -r -b %{FREEMEMLO} openwrt-atheros-2.6-vmlinux.lzma
Using default protocol (TFTP)
Raw file loaded 0x80040c00-0x80100bff, assumed entry at 0x80040c00
```

11. Se crea la imagen *flash* del kernel. Ser muy paciente en este paso.

```
RedBoot> fis create -r 0x80041000 -e 0x80041000 vmlinux.bin.l7
... Erase from 0xa8720000-0xa87e0000: ............
... Program from 0x80040c00-0x80100c00 at 0xa8720000: ............
... Erase from 0xa87e0000-0xa87f0000: .
... Program from 0x80ff0000-0x81000000 at 0xa87e0000: .
```

12. Se reinicia el kernel.

```
RedBoot>reset
Connection closed by foreign host.
```

Transcurridos unos minutos, cuando se vuelva a conectar la Fonera ya arrancará con OpenWrt (Figura C.8). Ahora sólo falta cambiar la contraseña de root, para poder entrar posteriormente con SSH, ya que al reiniciarla tiene un servidor Telnet en la IP 192.168.1.1.

```
>telnet 192.168.1.1[10]
```

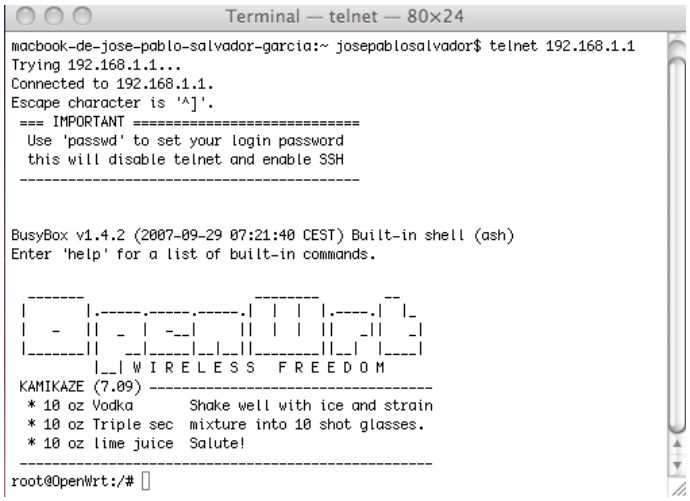

Figura C.8: Vista del directorio raíz del router Fonera con OpenWrt.

Para cambiar la contraseña y activar el acceso SSH, utilizar el siguiente comando:

```
>passwd root
```

Se solicitará introducir una clave y confirmarla, una vez hecho esto se tiene el router Fonera con acceso SSH habilitado. Al reiniciar al router se comprobará que el acceso mediante Telnet se ha desactivado y en su lugar se ha activado un servidor SSH.

[10] Hasta que no se cierre por completo la comunicación anterior el telnet no va a funcionar. Así que paciencia.

Apéndice D

Instalación de OpenWrt en el Linksys WRT54G

D.1. El router Linksys WRT54G

Linksys es un líder global reconocido en redes Ethernet, inalámbricas y VoIP para el usuario doméstico y pequeñas empresas. El modelo WRT54G es uno de los routers inalámbricos más comercializados, también sus variantes WRT54GL, con soporte para Linux, y WRT54GS, con mayor memoria RAM y *flash* y tecnología *SpeedBooster*[1] para aumentar la tasa de transferencia del modo G en un 30 %.

El router Linksys WRT54GL (Figura *D*.1) posee un conmutador o *switch* de 5 puertos IEEE 802.3 separados en dos VLANs[2] de uno y cuatro puertos (Figura *D*.2).

Figura D.1: Router Linksys WRT54GL.

En la Tabla D.1 se muestran las especificaciones del router WRT54GL:

[1] La tecnología *SpeedBooster* es un complemento compatible del estándar Wireless-G que aumenta el rendimiento real de la red inalámbrica hasta en un 35 %.

[2] VLAN: Red de área local virtual. Es una forma de crear redes lógicamente independientes dentro de una misma red física.

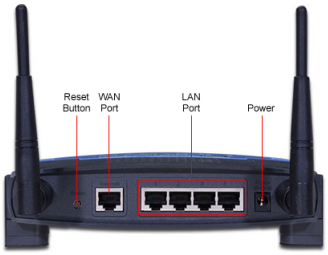

Figura D.2: Router Linksys WRT54GL. Vista posterior.

Modelo	WRT54G
Estándares	IEEE 802.3, IEEE 802.3u, IEEE 802.11g, IEEE 802.11b
Canales	11 canales (EEUU, Canadá); 13 canales (Europa, Japón)
Puertos/botones	Internet: Un puerto 10/100 RJ-45; LAN: Cuatro puertos 10/100 RJ-45 conmutados; Un puerto de alimentación; Un botón de reinicio; Un botón SecureEasySetup
Tipo de cables	Tipo CAT 5
Luces	Alimentación, DMZ, WLAN, LAN (1, 2, 3, 4), Internet, SecureEasySetup
Potencia de salida de radiofrecuencia	18 dBm
Cert./compat. UPnP	Compatible
Funciones de seguridad	Firewall con inspección exhaustiva de paquetes (SPI), directiva de Internet
Seguridad inalámbrica	*Wi-Fi Protected Access 2* o WPA2; (acceso Wi-Fi protegido 2), WEP, filtrado de direcciones MAC inalámbrico
Arquitectura	MIPS
Bootloader	CFE
System-On-Chip	Broadcom 5352EKPB
Velocidad de CPU	200 MHz
Flash-Chip	EON EN29LV302B-70TCP
Tamaño	*Flash*] 4 MB
RAM	16 MB
Wireless Broadcom	BCM43xx 802.11b/g (integrado)
Ethernet	*Switch* en CPU
USB	No
Serial	Sí
JTAG	Sí

Tabla D.1: Especificaciones técnicas del router Linksys WRT54GL

D.2. Proceso de cambio del *firmware*

Para llevar a cabo el cambio del *firmware* al router Linksys WRT54GL hay que elegir entre dos opciones de fichero:

- *.trx*
- *.bin*

Los ficheros *.trx* contienen el *firmware* en formato *raw*, es decir, exactamente como se escribirá en la memoria *flash*. Este formato es el que se utiliza cuando se actualiza el *firmware* o para una serie de dispositivos entre los que no se encuentra el router Linksys WRT54GL. En cambio el fichero *.bin* es igual que un fichero *.trx*, pero con una pequeña cabecera añadida al comienzo del fichero para que los modelos Linksys lo consideren válido.

A continuación se van a explicar tres métodos diferentes para la instalación de dicho *firmware* en este tipo de routers.

- A través de la interfaz *web* del Linksys. Esta es la forma más fácil de llevar a cabo la instalación. Para ello ha de disponerse el escenario mostrado en la Figura *D.3*.

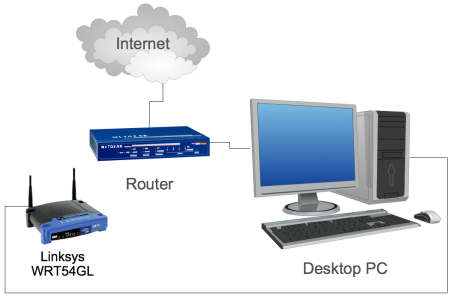

Figura D.3: Escenario 1 de cambio de *firmware* del router Linksys WRT54GL.

Los pasos que se deben realizar son:

1. Descargar el fichero de la imagen del *firmware* `openwrt-wrt54g-squashfs.bin` disponible en:

   `http://downloads.openwrt.org/kamikaze/7.09/brcm-2.4/openwrt-wrt54g-2.4-squashfs.bin`

   Nótese que se descarga la versión para el kernel 2.4, ya que para el kernel 2.6 no tiene soporte inalámbrico. Sin embargo en posteriores versiones de Kamikaze (v8.09) se resuelve dicho problema.
2. En el navegador introducir la dirección correspondiente al router:

   `http://192.168.1.1/`

   Escoger la opción de *Administration* y una vez dentro de ella la opción de actualizar el *firmware*, *Firmware Upgrade* (Figura *D.4*.).
3. Se carga la imagen del *firmware* `openwrt-wrt54g-squashfs.bin` y se presiona el botón de actualizar, *Upload*.
4. Se esperan alrededor de dos minutos, tras los cuales el router se reiniciará de forma automática quedando instalado el nuevo *firmware*.
5. Por último, una vez terminado, ya se puede acceder mediante telnet al router a través de la dirección IP 192.168.1.1, y comenzar la configuración personalizada del router como se comentará en el Apéndice D. En la Figura *D.5* se muestra la pantalla inicial del nuevo *firmware* en el router.

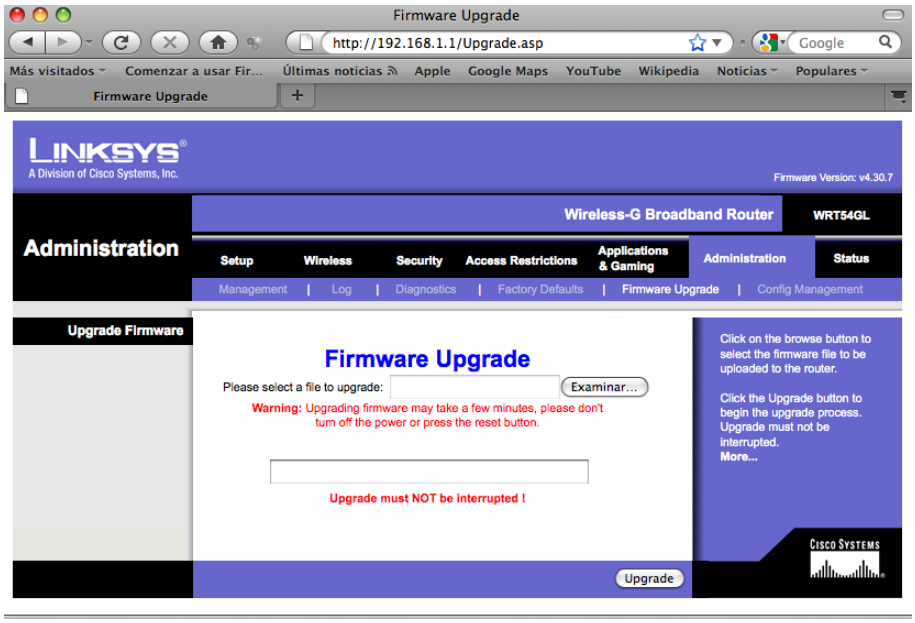

Figura D.4: Página de configuración del router Linksys WRT54GL.

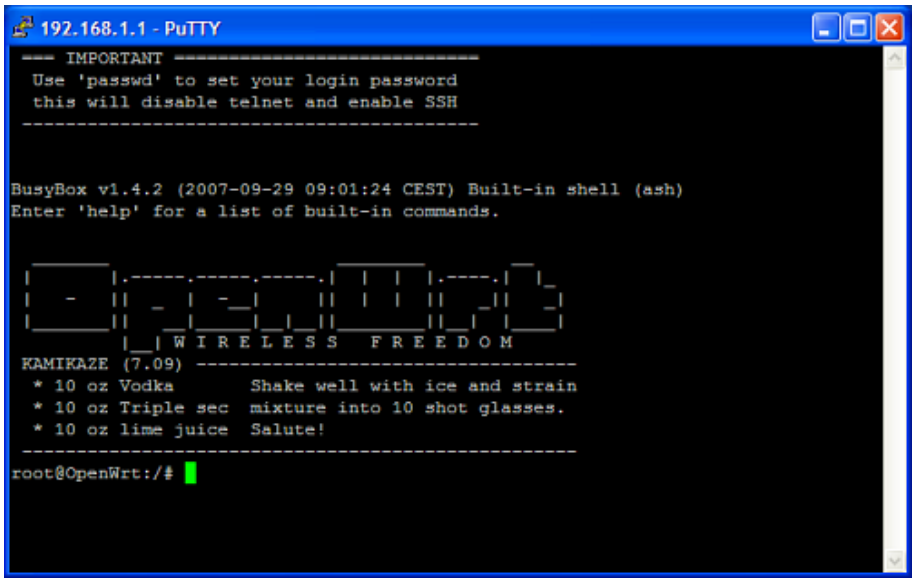

Figura D.5: Kamikaze en el router Linksys WRT54GL.

- Vía TFTP. El escenario es el mismo que se muestra en la Figura *D.3*. Los pasos que se deben seguir son:

  1. Se procede a conectar el router a un ordenador a través de cualquier puerto de su interfaz LAN, pero sin conectar el cable de alimentación de router. Se configura el ordenador con una dirección IP en el rango de 192.168.1.0, exceptuando la .1 y .255, ya que es el rango perteneciente a la dirección del router.
  2. A continuación se descarga en local la imagen de la versión del *firmware* que se desea instalar, *openwrt-wrt54g-squashfs.bin*.
  3. Se procede a lanzar el servidor TFTP a la dirección 192.168.1.1, indicando una serie de parámetros. La secuencia de comandos quedaría:

```
tftp 192.168.1.1
tftp> binary
tftp> trace
tftp> put openwrt-wrt54g-squashfs.bin
```

binary Activación del modo de transferencia binaria

trace Activación del modo de depuración para poder ver las trazas del programa

put Transferencia de la imagen del *firmware*

4. Posteriormente se alimenta el router. Tras unos segundos se podrá ver en la consola los asentimientos que se reciben a los paquetes de datos que envía la aplicación tftp.
5. Una vez que se haya transferido la imagen, el router se reiniciará automáticamente. Cuando el led de DMZ deje de parpadear y se apague completamente se podrá acceder al router con el nuevo *firmware* vía telnet cuya dirección IP será la 192.168.1.1.

- Usando el comando *mtd*:

  Esta opción permite actualizar el *firmware* habiendo ya instalado alguna versión. Para ello se ha de transferir la imagen del *firmware* a la carpeta temporal del router (/tmp) antes de realizar el *flasheo*. En este caso ha de disponerse el escenario mostrado en la Figura *D.6*.

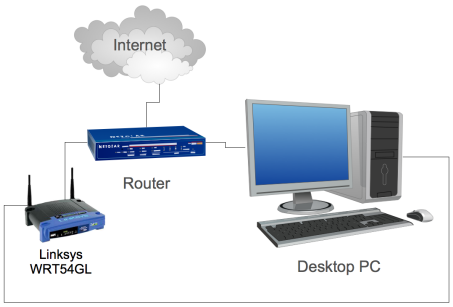

Figura D.6: Escenario 2 de cambio de *firmware* del router Linksys WRT54GL.

Los comandos que deberían ejecutarse son los siguientes:

```
cd /tmp/
wget http://downloads.openwrt.org/kamikaze/8.09/brcm-2.4/
openwrt-brcm-2.4-squashfs.trx
mtd write /tmp/openwrt-brcm-2.4-squashfs.trx linux
reboot
```

Apéndice E

Instalación y uso de la plataforma de desarrollo de OpenWrt en PC

E.1. Entorno de desarrollo del *firmware*

Como se ha comentado en el Apéndice B el router Fonera tiene escasa memoria, típico problema de los sistemas embebidos, por lo que es inviable instalar un compilador en ella, de ahí que sea necesario instalar un *Buildroot* o entorno de arranque en el ordenador para poder realizar el proceso conocido como cross-compilación, o compilación cruzada, en el que se necesita un nuevo compilador en otra máquina capaz de generar el código ejecutable para la arquitectura *mips*, específica del router.

En primer lugar se descargan los ficheros fuente de *Kamikaze*, que se encuentran en `http://downloads.openwrt.org/kamikaze/7.09/` (Figura *E*.1). En el caso de este proyecto se decide utilizar la versión 7.09 como se comenta en el Apéndice B. A partir de estos ficheros se construye la imagen.

Este repositorio contiene cuatro directorios principales:

- tools
- toolchain
- package
- target

Los directorios *tools* y *toolchain* contienen todo lo referente a las herramientas que se utilizarán para construir la imagen del *firmware*, el compilador y las librerías de C.

El resultado de compilar los ficheros fuente son tres nuevos directorios:

- tool_build
- toolchain_build_<arch>
- staging_dir_<arch>

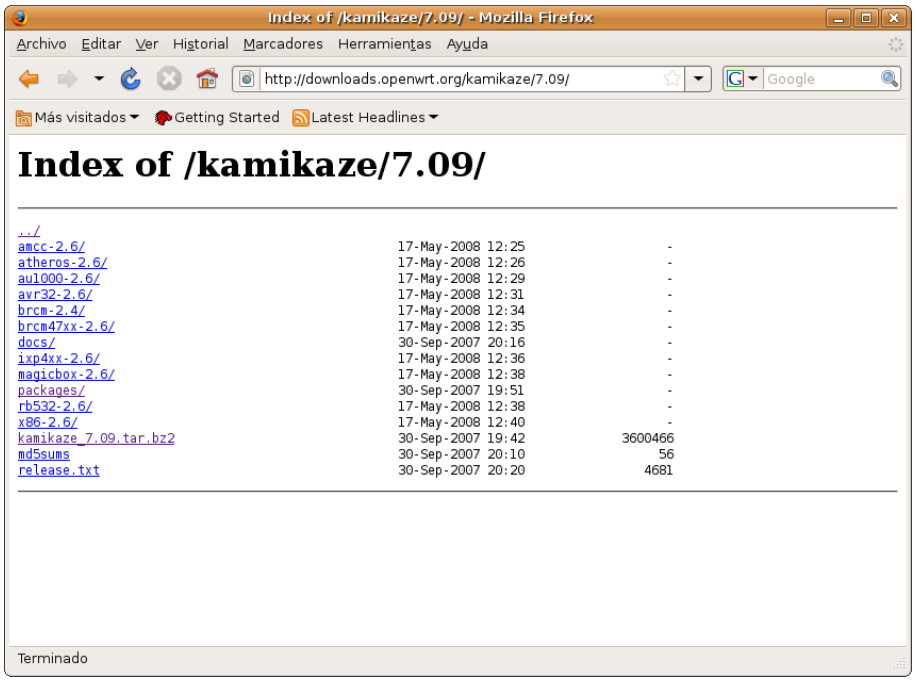

Figura E.1: Página de descarga del *firmware* OpenWrt Kamikaze.

Los directorios *tool_build* - directorio temporal- y *toolchain_build_<arch>* son directorios que contienen herramientas de desarrollo específicas de la arquitectura *objetivo*. El directorio *staging_dir_<arch>* contiene las herramientas resultantes, que se deberán utilizar para trabajar como si se estuviera en una máquina con la arquitectura *objetivo*, entre ellas, las herramientas de compilación cruzada. El marcador *<arch>* se corresponderá con la arquitectura del router, en este caso *mips*.

El directorio *package* contiene exactamente lo que indica su nombre, paquetes, que se podrán seleccionar para que también se compilen al crear la nueva imagen. Para ello las fuentes de los paquetes que básicamente consisten en un fichero de *makefile* y en un parche se encuentran en un repositorio distinto y que es necesario descargar de forma independiente.

En un *firmware* OpenWrt, casi todo es un .ipk, un paquete que puede ser añadido al *firmware* para proveerle de nuevas funcionalidades o ser eliminado para ahorrar espacio. Nótese que los paquetes se mantienen fuera de los ficheros fuente, pudiendo incluir paquetes que no vengan por defecto en la distribución descargándolos mediante subversión:

```
$ svn checkout https://svn.openwrt.org/openwrt/packages packages
```

El directorio *target* contiene las herramientas específicas de una plataforma embebida. Un directorio que cabe destacar es *target/linux*, que contiene todos los parches y perfiles de configuración para cada una de las arquitecturas soportadas.

E.2. Construyendo la imagen del *firmware*

En este apartado se van a detallar los pasos necesarios para construir una imagen de OpenWrt que incluya aquellas funcionalidades necesarias para el correcto funcionamiento

de la implementación de NEMO BS dadas las características del router Fonera.

En primer lugar se ejecuta el comando `make prereq` para comprobar qué paquetes faltan. A continuación se actualiza la lista de paquetes: `sudo apt-get update`. Ahora habría que instalar los paquetes para poder cumplir todas las dependencias, como por ejemplo:

```
sudo apt-get install g++
sudo apt-get install patch
sudo apt-get install flex
sudo apt-get install gibbon
...
```

Una vez instalados todos los paquetes necesarios se ha de ejecutar el comando `make menuconfig` que hará aparecer la interfaz de configuración que se muestra en la Figura *E.2*.

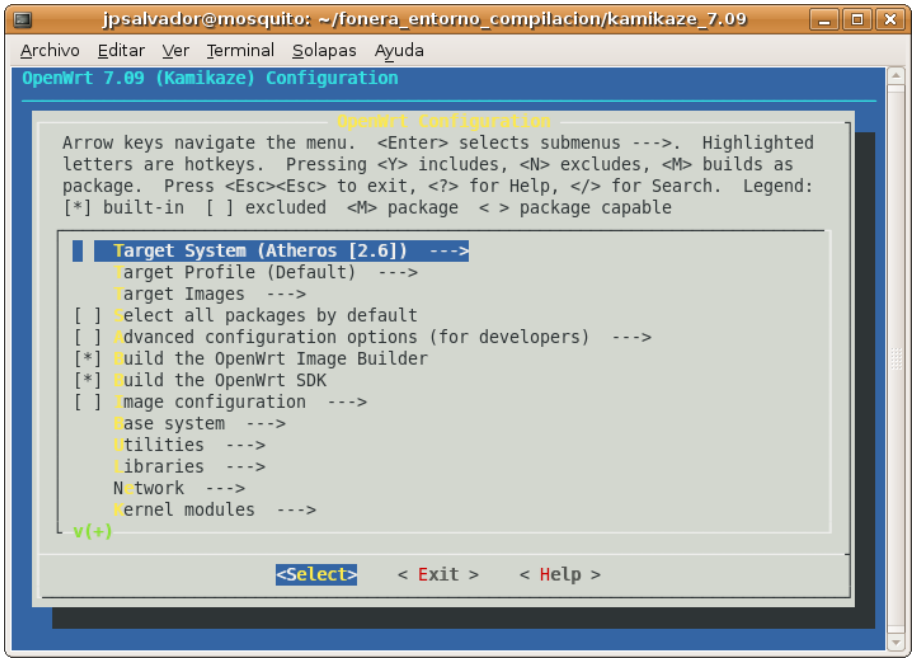

Figura E.2: Interfaz de configuración de OpenWrt Kamikaze.

En el menú aparecerán distintas opciones para ser seleccionadas. Casi todas las opciones permiten elegir entre *y/m/n* que representan:

- <*>: (Pulsando y) Esta opción será incluida en la imagen del *firmware*.
- <M>: (Pulsando m) Esta opción será compilada pero no incluida, para su posterior instalación.
- <>: (Pulsando n) Esta opción no será compilada.

Especialmente importante es seleccionar en la sección *Target System* el kernel que se quiere instalar. En el caso de este proyecto se ha optado por un kernel 2.6 para *Atheros*, debido a que tiene que ser compatible con la características de la Fonera.

Por otro lado, resulta conveniente seleccionar las opciones *Build the OpenWrt Image Builder* y *Build OpenWrt SDK*. Ambas ayudarán a poder instalar estas herramientas, propias de la imagen creada, en otros PCs. La opción *Build the OpenWrt Image Builder* permite construir una imagen personalizada del *firmware* OpenWrt, mientras que *Build OpenWrt SDK*[1] crea un entorno de desarrollo de *software*, lo que permite no tener un *Buildroot*. El SDK es una versión recortada del *Buildroot*, que incluye el *toolchain*, y todas las librerías y ficheros requeridos para croscompilar aplicaciones para OpenWrt.

También se ha de seleccionar el módulo del kernel correspondiente al soporte de red para IPv6 *kmod-ipv6*. Otros módulos también son importantes, pero al seleccionar las opciones mencionadas al comienzo de este párrafo la mayor parte de los paquetes son seleccionados.

Tras completar la configuración, se procede a salir guardando los cambios. Ahora hay que ejecutar la orden `make` o `make V=99` para compilar la nueva imagen del kernel con los cambios realizados. La segunda orden permite tener un mayor nivel de información sobre el proceso de compilación.

Durante este proceso, el cargador de arranque descargará todas las fuentes al directorio /dl/ y comenzará a colocarlas y compilarlas en el directorio *build_ <arch>*. Al terminar, el *firmware* se encontrará en el directorio */bin* y los paquetes en el directorio */bin/packages*.

Aún no se ha terminado, habrá que recompilar esa imagen, para poder tener el módulo que da soporte a túneles en IPv6 (*ip6_ tunnel.ko*). Para ello habrá que modificar la variable que genera dicho módulo del fichero .config que se encuentra dentro de `<personal_directory>/kamikaze_7.09/build_mips/linux-2.6-atheros/linux-2.6.21.5`:

```
CONFIG_IPV6_TUNNEL=m
```

Una vez modificado se vuelve a ejecutar la orden `make`, obteniendo el mencionado módulo.

E.3. Compilando aplicaciones para OpenWrt Kamikaze

Una vez generada la imagen de OpenWrt con los requerimientos necesarios, se procede a compilar el *software* que implementa el protocolo NEMO BS en el caso de los routers móviles. Como ya se ha comentado en otros apéndices, no era posible instalar un entorno de desarrollo en el router Fonera, para ello habrá que utilizar herramientas de compilación cruzada. Tras compilar la imagen se crean nuevos directorios, uno de ellos */staging_ build_ mips/* contiene todas las herramientas de compilación. Estos compiladores se encuentran dentro de la carpeta */staging_ build_ mips/bin/*.

Ahora para poder compilar la aplicación es necesario establecer el *path* o ruta del compilador:

```
$ export PATH=<personal_directory>/kamikaze_7.09/staging_dir_mips/bin/:$PATH
```

Por último, tan sólo queda compilar el *software*. De entre todos los compiladores que

[1] SDK: *Software Development Kit*

se disponen, se elige *mips-linux-uclibc*[2]*-gcc*. Finalmente la compilación se realizara de la siguiente forma:

```
$ mips-linux-uclibc-gcc *.c -lm -o poseidonMR
```

En realidad tampoco es necesario saber esto, ya que para compilar el *software* será suficiente con ejecutar la orden `make` de cada una de las implementaciones (agente local y router móvil).

[2] *uclibc*: librería de C optimizada para sistemas embebidos.

Apéndice F

Montaje y configuración del escenario desplegado

F.1. Escenario desplegado

En la Figura *F.1* se presenta el escenario en que se ha comprobado y evaluado la implementación desarrollada. En ella se muestran los diferente dispositivos empleados en la implementación.

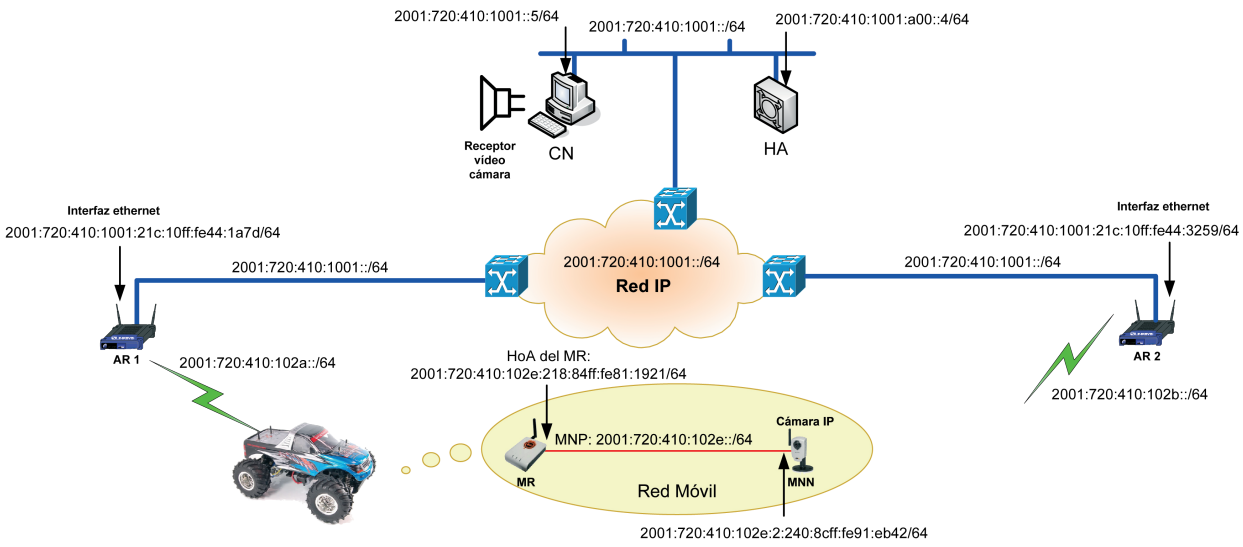

Figura F.1: Escenario de evaluación para el protocolo NEMO BS.

La descripción de cada uno de los dispositivos es:

- Ordenadores. Son empleados como agente local y nodo corresponsal. No necesitan una configuración especial, el agente local debe ejecutar el *software* para la implementación de NEMO BS en la red hogar, y el nodo corresponsal tan sólo tener configurados las rutas a los distintos dispositivos.
- Routers Linksys WRT54GL. Se utilizan como routers de acceso.
- Routers Fonera 2200. Se utilizan como routers móviles.

- Cámara Axis 207W. Se utiliza como nodo móvil, MN.

F.2. Configuración de los equipos

A continuación se describe con mayor detalle la configuración de cada uno de los dispositivos que conforman la red.

F.2.1. Nodo Corresponsal, CN

El equipo utilizado como nodo corresponsal es un portátil Asus Eee PC 1000HE (Figura *F.2*). Este dispositivo cuenta con el sistema operativo Linux, distribución Debian 2.6.26. Para facilitar la validación del funcionamiento de la aplicación se han instalado una serie de paquetes:

- tcpdump
- wireshark

Figura F.2: Portátil que actúa como nodo corresponsal.

Además para poder visualizar el *streaming* de la cámara IP se han instalado una serie de paquetes, siendo el reproductor multimedia de código libre VLC (Figura *F.3*), perteneciente al proyecto VideoLAN[1], el que se ha utilizado como cliente de vídeo usando RTSP, HTTP o UDP. Otros clientes utilizados para poder visualizar el vídeo han sido los navegadores Mozilla Firefox y Konqueror.

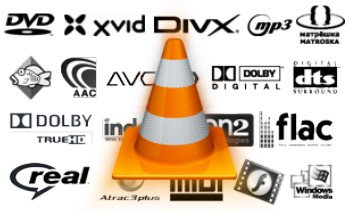

Figura F.3: Logo del reproductor multimedia VLC.

[1] VideoLAN *project*: www.videolan.org/vlc/

En primer lugar se ha de comprobar que el módulo de IPv6 se encuentra cargado, posteriormente deben configurarse los parámetros de IPv6, concretamente las rutas desde el CN al resto de dispositivos de la red. Para que dicha configuración se realice al arrancar el equipo se ha creado un *script* en */etc/init.d/* que ha sido enlazado a */etc/rc*. A continuación se detalla la configuración realizada en el *script*:

```
sudo ip -6 ro add 2001:720:410:102b::/64 via 2001:720:410:1001:21c:10ff:fe44:3259
dev eth0
sudo ip -6 ro add 2001:720:410:102a::/64 via 2001:720:410:1001:21c:10ff:fe44:1a7d
dev eth0
sudo ip -6 ro add 2001:720:410:102e::/64 via 2001:720:410:1001:0a00:0:0:4
dev eth0
```

F.2.2. HA

El equipo utilizado como agente local es un portátil Dell Latitute D600 (Figura *F.4*). Este dispositivo cuenta con el sistema operativo Linux, distribución Ubuntu 8.10. Para facilitar la validación del funcionamiento de la aplicación se han instalado una serie de paquetes:

- tcpdump
- wireshark

Figura F.4: Portátil que actúa como agente local.

En cuanto a la configuración del agente local, no se ha instalado ningún paquete especial ni tampoco ha sido necesario ningún fichero o *script* de configuración inicial. Este dispositivo tan sólo tiene instalado el *software* perteneciente al agente local y un fichero de configuración con la lista de sus routers móviles correspondientes.

F.2.3. Routers móviles

Como se ha mencionado ya anteriormente el router Fonera 2200 es el encargado de actuar como router móvil. En primer lugar se modifica el fichero de configuración de la red */etc/config/network*. Se crea una interfaz inalámbrica y se rompe el *bridge* que viene por defecto en los routers Fonera 2200.

```
# Copyright (C) 2006 OpenWrt.org

config interface loopback
        option ifname   lo
        option proto    static
        option ipaddr   127.0.0.1
        option netmask  255.0.0.0

config interface lan
        option ifname   eth0
        #option type    bridge
        option proto    static
        option ipaddr   192.168.0.3
        option netmask  255.255.255.0
        #option gateway  163.117.140.2
        #option dns      163.117.140.2

config interface wifi
        option ifname   ath0
        option proto    static
        option ipaddr   192.168.5.30
        option netmask  255.255.255.0
```

Se salvan los cambios y se reinicia para que la configuración se lleve a cabo:

```
/etc/init.d/network restart
```

Para el correcto funcionamiento es necesario la instalación de una serie de módulos:

- radvd[2]
- ip
- kmod-ipv6.ko
- kmod_ip6tables.ko
- 6tunnel.ko
- ip6_tunnel.ko

El primero de los paquetes a instalar, *radvd* es el demonio para el envío de los anuncios de router o *Router Advertisements*, necesarios para que el router móvil detecte el cambio de red y se conecte al correspondiente router de acceso. A continuación se instala el paquete con herramientas para gestionar la configuración de la red, *ip*, el que da soporte IPv6 (*kmod-ipv6*) y otro paquete que da soporte a la gestión del *firewall* en IPv6 (*kmod-ip6tables*). Los últimos dos paquetes hacen posible el uso de túneles en IPv6.

OpenWrt utiliza un sistema de gestión de paquetes denominado *ipkg*[3].

[2] http://www.litech.org/radvd/

[3] En primer lugar se modifica la primera línea del fichero */etc/ipkg.conf* ya que por defecto la dirección de la cual se descargan los paquetes es errónea:
`src release http://downloads.openwrt.org/kamikaze/7.09/atheros-2.6/packages`

En líneas generales, resulta muy similar a *apt-get*, pero es más ligero, y por tanto más adecuado para este tipo de dispositivos. Para llevar a cabo la instalación de dichos paquetes se han de ejecutar los siguientes comandos:

```
OpenWrt:/# ipkg update
OpenWrt:/# ipkg upgrade
OpenWrt:/# ipkg install radvd
OpenWrt:/# ipkg install ip
OpenWrt:/# ipkg install kmod-ipv6
OpenWrt:/# ipkg install kmod-ip6tables
```

Tras la instalación de estos paquetes se carga el módulo IPv6, para que se cargue siempre al arrancar la Fonera: `insmod ipv6`. A continuación, se obtienen los otros dos paquetes de la imagen creada del *firmware* OpenWrt. Se encuentran en el directorio:
`<personal_directory>/kamikaze_7.09/build_mips/linux-2.6-atheros/linux-2.6.21.5/net/ipv6/`

Se copian en la Fonera mediante *netcat* en la carpeta `/lib/modules/2.6.21.5/`. Después de esto, se deben cargar estos módulos al iniciar la Fonera, para ello se completa el módulo que carga el soporte de IPv6 añadiéndolos. El fichero que se llama *20-ipv6* y se encuentra dentro de */etc/modules.d/* queda finalmente:

```
ipv6
tunnel6
ip6_tunnel
```

Es interesante instalar otros paquetes como *tcpdump* que permite analizar el tráfico que circula por la red, e *iperf*[4], el cual es una utilidad que permite medir el ancho de banda y rendimiento de una conexión entre dos equipos. Además puede crear flujos de datos TCP y UDP. Por tanto, se trata de una herramienta cliente-servidor.

```
OpenWrt:/# ipkg install tcpdump
OpenWrt:/# ipkg install iperf
```

Para evitar posibles conflictos o problemas, se deshabilitan el *firewall* y los servidores de *DNS* y *HTTP* de la siguiente forma:

```
OpenWrt:/# /etc/init.d/firewall stop
OpenWrt:/# rm /etc/init.d/firewall
OpenWrt:/# /etc/init.d/dnsmasq stop
OpenWrt:/# rm /etc/init.d/dnsmasq
OpenWrt:/# /etc/init.d/httpd stop
OpenWrt:/# rm /etc/init.d/httpd
```

Se configura también la interfaz inalámbrica del router de acuerdo al escenario desplegado. Estos cambios se realizan en el fichero */etc/config/wireless*

[4] http://sourceforge.net/projects/iperf/

```
config wifi-device  wifi0
        option type     atheros
        option channel  4
        option ap       00:1c:10:44:1a:7f

config wifi-iface
        option device   wifi0
        option network  lan
        option mode     sta
        option ssid     poseidon
        option encryption none
```

El router móvil a su vez también envía *Routers Advertisements* a los nodos móviles pertenecientes a su red, para ello se modifica el fichero */etc/radvd.conf*

```
#
# radvd configuration generated by radvdump 1.0
#

interface eth0
{
        AdvSendAdvert on;
        # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvIntervalOpt on;
        # These settings cause advertisements to be sent every 3-10
        # seconds. This range is good for 6to4 with a dynamic IPv4 address,
        # but can be greatly increased when not using 6to4 prefixes.
        MinRtrAdvInterval 2;
        MaxRtrAdvInterval 5;
        #AdvManagedFlag off;
        #AdvOtherConfigFlag off;
        #AdvReachableTime 0;
        #AdvRetransTimer 0;
        #AdvCurHopLimit 64;
        AdvHomeAgentFlag off;
        #AdvDefaultPreference high;
        #AdvLinkMTU 1476;
        #AdvSourceLLAddress on;

        prefix 2001:720:410:102e::/64
        {
                #AdvValidLifetime 2592000;
                #AdvPreferredLifetime 604800;
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;#by default to off
        };# End of prefix definition
}; # End of interface definition
```

Por último, se crea un *script* de inicio en */etc/init.d* en el que se configuran los parámetros IPv6, se habilita el demonio *radvd* y el inicio del *software* que implementa NEMO BS de forma automática al arrancar el router, consiguiéndose total autonomía:

```
##########/etc/init.d/poseidon#######################
#!/bin/sh /etc/rc.common
# Copyright (C) 2006 OpenWrt.org
START=60

start() {
        wlanconfig ath0 destroy
        wlanconfig ath0 create wlandev wifi0 wlanmode sta
        /etc/init.d/radvd start
        ip link set ath0 up
        iwconfig ath0 essid poseidon channel 4 ap 00:1c:10:44:1a:7f
        ip -6 addr add 2001:720:410:102a:218:84ff:fe80:a015/64 dev ath0
        ip -6 addr add 2001:720:410:102e:218:84ff:fe80:a014/64 dev eth0
        if [ -e /poseidon/AUTOINIT ]; then
                /poseidon/poseidon -d &
                sh /poseidon/handover.sh &
        fi
}

stop() {
        killall poseidon
}
```

A continuación, se le dan permisos de ejecución con *chmod*. Finalmente se crea un enlace a dicho fichero desde el directorio */etc/rc.d* (demonio de arranque) para que así se inicie este *script* en el arranque del router:

```
OpenWrt:/etc/rc.d# ln -s ../init.d/poseidon S60poseidon
```

F.2.4. Routers de acceso

Al igual que en el caso de los routers móviles, en primer lugar se modifica el fichero de configuración de la red */etc/config/network*. Se ha roto el *bridge* entre los interfaces LAN y WAN que viene por defecto configurado en los routers Linksys WRT54G. En ese mismo fichero también se han creado dos interfaces Ethernet, mediante la configuración de VLANs en el *switch* del router. Además se crea una interfaz inalámbrica.

```
#### VLAN configuration
config switch eth0
    option vlan0    "0 1 2 5*"
    option vlan1    "3 5"
    option vlan2    "4 5"

#### Loopback configuration
config interface loopback
```

```
    option ifname   "lo"
    option proto    static
    option ipaddr   127.0.0.1
    option netmask  255.0.0.0

#### LAN configuration
config interface lan
    option ifname   "eth0.0"
    option proto    static
    option ipaddr   192.168.2.1
    option netmask  255.255.255.0

#### LAN2 configuration
config interface lan2
    option ifname   "eth0.1"
    option proto    static
    option ipaddr   192.168.1.1
    option netmask  255.255.255.0

#### WAN configuration
config interface    wan
    option ifname   "eth0.2"
    option proto    dhcp

#### WIFI configuration
config interface wifi
    option ifname   wl0
    option proto    static
    option ipaddr   192.168.3.1
    option netmask  255.255.255.0
```

Tras crear la interfaz inalámbrica, se modifican una serie de parámetros de la misma en su fichero de configuración (*/etc/config/wireless*) como, por ejemplo, el canal de funcionamiento, el modo en que trabaja o el nombre de la red. A continuación se muestra este fichero:

```
config wifi-device  wl0
    option type     broadcom
    option channel  4

config wifi-iface
    option device   wl0
    option network  wifi
    option mode     ap
    option ssid     poseidon
    option encryption none
```

Para el correcto funcionamiento de la aplicación es necesario instalar una serie de módulos:

- radvd
- ip
- kmod-ipv6.ko
- kmod_ip6tables.ko
- wl

El soporte que dan estos módulos ya se ha comentado en la sección F.2.3. El comando *wl* se utiliza para la configuración específica de dispositivos con chipset *Broadcom*. Nuevamente *ipkg* es el sistema usado para la gestión de los paquetes:

```
OpenWrt:/# ipkg update
OpenWrt:/# ipkg upgrade
OpenWrt:/# ipkg install radvd
OpenWrt:/# ipkg install ip
OpenWrt:/# ipkg install kmod-ipv6
OpenWrt:/# ipkg install kmod-ip6tables
OpenWrt:/# ipkg install tcpdump
OpenWrt:/# ipkg install wl
```

Al igual que en el caso de los routers móviles, para evitar posibles conflictos o problemas, se inhabilita el *firewall* y los servidores de *DNS* y *HTTP* de la siguiente forma:

```
OpenWrt:/# /etc/init.d/firewall stop
OpenWrt:/# rm /etc/init.d/firewall
OpenWrt:/# /etc/init.d/dnsmasq stop
OpenWrt:/# rm /etc/init.d/dnsmasq
OpenWrt:/# /etc/init.d/httpd stop
OpenWrt:/# rm /etc/init.d/httpd
```

A continuación se configura el demonio de *radvd* (*/etc/radvd.conf*), para posibilitar el envío de *Router Advertisements*, para que el router anuncie el prefijo correspondiente de su red según la interfaz.

```
#
# radvd configuration generated by radvdump 1.0
# based on Router Advertisement from fe80::205:1cff:fe0f:354b
# received by interface wl0
#

interface wl0
{
    AdvSendAdvert on;
    # Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
    #AdvIntervalOpt on;
    # These settings cause advertisements to be sent every 0.5-1
    # seconds. This range is good for 6to4 with a dynamic IPv4 address,
```

```
    # but can be greatly increased when not using 6to4 prefixes.
    MinRtrAdvInterval 0.03;
    MaxRtrAdvInterval 0.07;
    #AdvManagedFlag off;
    #AdvOtherConfigFlag off;
    #AdvReachableTime 0;
    #AdvRetransTimer 0;
    #AdvCurHopLimit 64;
    #AdvHomeAgentFlag off;
    #AdvDefaultPreference high;
    #AdvLinkMTU 1476;
    #AdvSourceLLAddress on;
    prefix 2001:720:410:102a:21c:10ff:fe44:1a7f/64
    {
        #AdvValidLifetime 2592000;
        #AdvPreferredLifetime 604800;
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;#by default to off
    };# End of prefix definition
}; # End of interface definition
```

Por último, se crea un *script* de inicio en */etc/init.d* en el que se configuran los parámetros IPv6, la potencia de transmisión del router se establece a 1 mW:

```
##########/etc/init.d/poseidon######################
#!/bin/sh /etc/rc.common
# Copyright (C) 2006 OpenWrt.org
# ip -6 addr add 2001:720:410:1001::/64
START=60

start() {
        wl txpwr1       1 #It changes the power settings to X mW (or X dBm)
        ip -6 addr add 2001:720:410:102a:21c:10ff:fe44:1a7f/64 dev wl0
        ip -6 addr add 2001:720:410:1001:21c:10ff:fe44:1a7d/64 dev eth0.0
        #ip -6 addr add fd33:3333:3333:0001::2/64 dev eth0.0
        #ip -6 ro add 2001:720:410:102b::/64 via fd33:333:3333:0001::2 dev eth0.0
        #ip -6 ro add default via fd33:3333:3333:0001::1 dev eth0.0
        ip -6 ro add 2001:720:410:102b::/64 via 2001:720:410:1001:21c:10ff:fe44:3259
         dev eth0.0
        ip -6 ro add default via 2001:720:410:1001:21c:10ff:fe44:1a7d dev eth0.0
}

stop() {
        killall poseidon
}
```

A continuación, se le dan permisos de ejecución con *chmod*. Finalmente se crea un enlace a dicho fichero desde el directorio */etc/rc.d* para que así se inicie este *script* en el arranque del router:

```
OpenWrt:/etc/rc.d# ln -s ../init.d/poseidon S60poseidon
```

Por último, se crea otro fichero que se encarga de iniciar el demonio *radvd* y configurar la opción de *forwarding* o reenvío de paquetes para que actúe como router:

```
#!/bin/sh /etc/rc.common
# Copyright (C) 2006 OpenWrt.org
START=50

start() {
        echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
        mkdir -p /var/log
        mkdir -p /var/run
        /usr/sbin/radvd
}

stop() {
        killall radvd
        echo 0 > /proc/sys/net/ipv6/conf/all/forwarding
}
```

Al igual que el otro *script* habrá que enlazarlo al *rc* (demonio de arranque), para que se ejecute dicho *script* en el inicio:

```
OpenWrt:/etc/rc.d# ln -s ../init.d/radvd S50radvd
```

Apéndice G

Instalación y configuración del *software* desarrollado

En este apéndice se van a explicar los pasos necesarios para instalar el *software* en el agente local, HA, y en el router móvil, MR.

Una vez configurados los routers de acceso, router móviles y equipos se puede comenzar a utilizar dicho *software*. En primer lugar, para poder probar la implementación NEMO BS, habrá que instalarla tanto en el agente local como en los distintos routers móviles.

Hay que disponer del paquete *poseidon_mr* o *poseidon_ha*, según el dispositivo. A continuación se descomprime el paquete situándose en el directorio raíz de la carpeta del *software* y se ejecuta la orden `make` para compilarlo. Por ejemplo, para el caso de un router móvil el conjunto de comandos sería el siguiente:

```
root@MR1:~# tar -xvvf poseidon_mr.gz
root@MR1:~# move poseidon_mr /
root@MR1:/# cd /poseidon_mr
root@MR1:~/poseidon_mr# make
```

Tras compilar la aplicación, se creará en el directorio un fichero ejecutable (*poseidon_mr* o *poseidon_ha*) al cual hay que invocar para proceder a la ejecución del *software*.

También ha de decidirse a la hora de ejecutar el nivel de detalle de los mensajes que aparecerán por pantalla. Existen cuatro tipos que se muestran a continuación, estando ordenados de mayor a menor nivel de detalle:

- LOG_DEBUG. Mensajes que contienen información para la depuración del programa.
- LOG_INFO. Mensajes de información general del programa.
- LOG_WARNING. Mensajes de alerta del programa.
- LOG_ERR. Mensajes de error del programa.

Cabe notar que emplear un nivel de detalle minucioso, caso del LOG_DEBUG, permite tener un mejor seguimiento del ciclo de ejecución del *software*, mostrando todos los

mensajes de configuración, de información, de aviso y de error. Mientras que por ejemplo el nivel LOG_ERR apenas proporciona información, sólo muestra los mensajes de error, en el caso de que se produzca alguno, terminando posteriormente la ejecución.

Para la implementación de estos mensajes se ha empleado el estándar *syslog*. Es un protocolo para el envío de mensajes de registro en una red IP, que hoy en día está presente por defecto en casi todos los sistemas Unix y Linux, existiendo también implementaciones para otros sistemas operativos como Microsoft Windows. El mensaje de registro suele contener información sobre la seguridad del sistema aunque puede incluir cualquier información, por ejemplo:

- Variaciones en el funcionamiento normal del sistema.
- Alertas cuando ocurre algún evento.
- Información sobre las actividades del sistema operativo.
- Errores de *hardware* o *software*.

Junto con cada mensaje se incluye la hora y la fecha del envío. También es posible registrar el funcionamiento normal de un programa. Concretamente para el desarrollo de esta opción se emplea una función incluida en C que envía mensajes al demonio *syslog*, el cual escribe el mensaje al correspondiente registro del sistema, y lo reenvía a una lista de usuarios o a alguna unidad de registro en otra parte de la red.

En primer lugar se emplea la siguiente función para abrir el registro de los mensajes:

```
openlog("NEMO SW", LOG_PID|LOG_PERROR, LOG_DAEMON);
```

De esta forma se escriben los mensajes tanto en la salida estándar como en el registro del sistema.

Para obtener los distintos niveles de detalle se establece una prioridad a cada categoría de mensajes empleando la función:

```
int logmask = setlogmask(LOG_UPTO(priority));
```

Por último, para escribir los distintos mensajes se realiza una llamada a la función de C *syslog*, indicando como primer parámetro la categoría del mensaje y como segundo el mensaje a imprimir. Un ejemplo de esta llamada sería la siguiente:

```
syslog(LOG_DEBUG, "DEB Opening binding cache file saved in /tmp");
```

Este nivel de detalle ha de especificarse como el primer parámetro de entrada a la hora de ejecutar el *software*:

```
root@MR1:~/poseidon_mr# ./poseidon_mr LOG_XXXX
```

Siendo XXXX, DEBUG o INFO o WARNING o ERR. En la Figura G.1 se muestra la salida por pantalla de la aplicación para el mayor nivel de detalle posible.

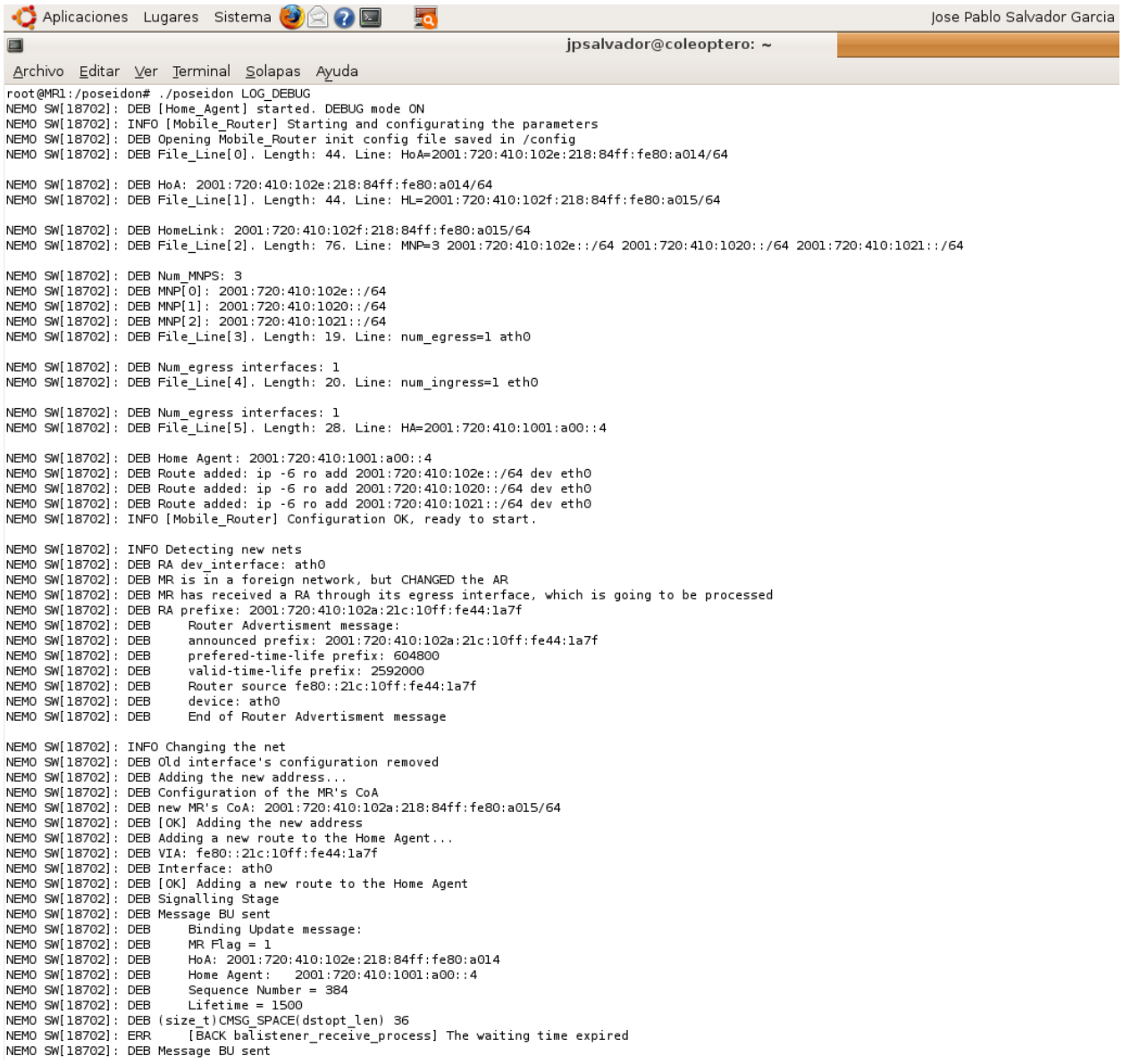

Figura G.1: Salida por pantalla de la aplicación NEMO BS.

Apéndice H

Streaming de vídeo

H.1. *Streaming.* Introducción

Antes de que el *streaming* apareciera, la reproducción de contenidos multimedia a través de Internet implicaba necesariamente descargar los archivos completamente, lo que dado el tamaño de este tipo de archivos conllevaría un tiempo de espera considerable. El *streaming* permite reproducir los archivos al mismo tiempo que se produce la descarga, lo que además del ahorro en tiempo, abre un nuevo abanico de aplicaciones.

H.2. Protocolos para el transporte de vídeo

Para el *streaming* se pueden utilizar protocolos llamados "ligeros", como UDP y RTSP, que permiten la entrega de paquetes de datos con una velocidad mucho mayor que las obtenidas por TCP y HTTP, debido a que cuando TCP y HTTP sufren un error de transmisión, siguen intentando transmitir los paquetes de datos perdidos hasta conseguir una confirmación de que la información llegó en su totalidad. Sin embargo, UDP resulta más adecuado porque continúa mandando los datos sin tener en cuenta interrupciones. En una aplicación multimedia estas pérdidas no son críticas, además de que la espera de la confirmación y el manejo de los errores de transmisión o de la pérdida de algún paquete tiene peores consecuencias para una aplicación de este tipo.

RTSP fue específicamente diseñado para el *streaming* en la red. Sin embargo, en este proyecto no se ha podido utilizar este protocolo porque los clientes de vídeo aún no tienen soporte IPv6 para RTSP, por lo que la transmisión desde la cámara Axis se ha apoyado en UDP y HTTP (este último para su visualización a través de un navegador web).

H.3. Clientes para el *streaming*

Previamente a decidir en qué protocolos se basaría la transmisión, se realizó una evaluación de los distintos clientes de vídeo

- VLC. Este cliente es de código libre, convirtiéndolo en una opción deseada. Sin embargo, este reproductor no soporta RTSP sobre IPv6 debido a la librería que utiliza

(*Live.com*), la cual no soporta IPv6 a la hora de escribir. Aún así, dicho reproductor se va a utilizar con el protocolo UDP y HTTP para la recepción de vídeo.

- TOTEM. Este cliente también de código libre, no tiene tampoco soporte par RTSP sobre IPv6.
- Real Player 11. No tenía soporte de RTSP sobre IPv6 para el sistema operativo Linux.
- Otros clientes probados pero que tampoco cumplían los requerimientos son: MPlayer, XMMS, Zinf.

H.4. Cámara IP Axis 207W

En primer lugar el acceso a la cámara IP no se va a poder realizar a través de su dirección IPv6 ya que esta opción se encuentra desactivada por defecto. Por lo que habrá que configurar la dirección IPv4 de la cámara utilizando el comando *arp*. Antes de nada, habrá que configurar la dirección de la interfaz del equipo, por ejemplo: `192.168.0.5`. A continuación utilizando la opción *-s* del comando *arp* se le asignará una dirección a la cámara en el rango de la dirección IP del equipo al que se conecta. Con la opción *temp* se evita que se añada dicha dirección de forma permanente a la cache ARP:

```
arp -s <Dirección IP> <Número de serie> temp
```

En el caso de este PFC, la configuración realizada es la siguiente:

```
arp -s 192.168.0.7 00:40:8c:9c:cb:fc temp
```

Para comprobar que existe conectividad a la cámara, se hace uso del comando *ping*. En el caso de que no hubiese respuesta, se procedería a desconectar la cámara y volver a conectar la corriente para reiniciarla. Una vez que se obtenga respuesta de la cámara, se accede a ella a través del navegador:

```
http://192.168.0.7
```

La primera vez que se accede al dispositivo aparecerá un cuadro de diálogo que solicitará que se configure la contraseña de *root* (Figura *H*.1).

En ocasiones posteriores al acceder a la cámara vía HTTP se solicitará el nombre y contraseña del usuario (Figura *H*.2).

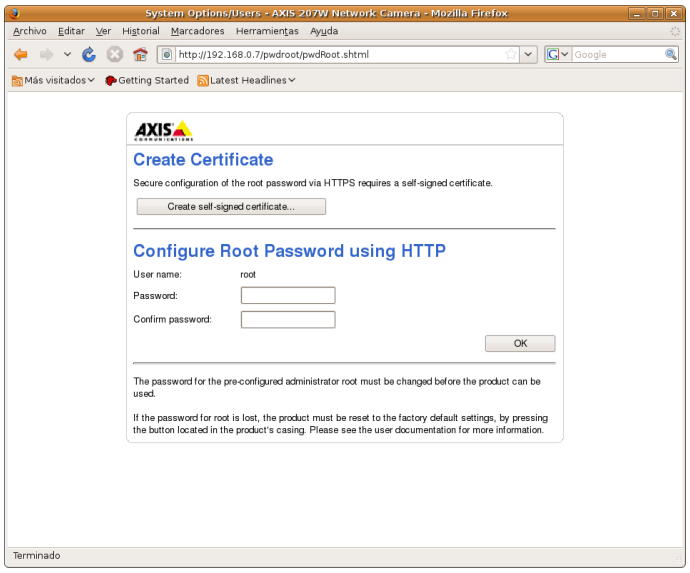

Figura H.1: Cámara IP AXIS 207W. Proceso de autenticación.

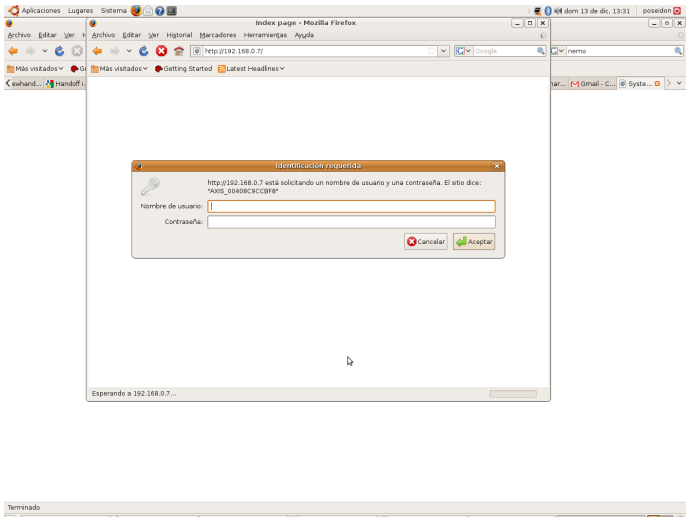

Figura H.2: Cámara IP AXIS 207W. Identificación requerida al acceder.

Ahora se procede a habilitar el uso de IPv6, ya que por defecto se encuentra desactivado como se muestra en la Figura *H.3*.

Con esto se consigue poder acceder a la cámara a través de su dirección IPv6. Al acceder vía HTTP, como dirección URL, se debe poner la dirección IPv6 entre corchetes [BLFM05], como por ejemplo:

`http://root:pass@[2001:5c0:84d9:2:240:8cff:fe6b:3cb9]:80`[1]`/view/index.shtml`

También se debe comprobar que está seleccionada la opción del protocolo RTSP, para poder habilitar el *streaming* de vídeo con formato MPEG-4 (Figura *H.4*).

[1] 80: Puerto de servicio para HTTP

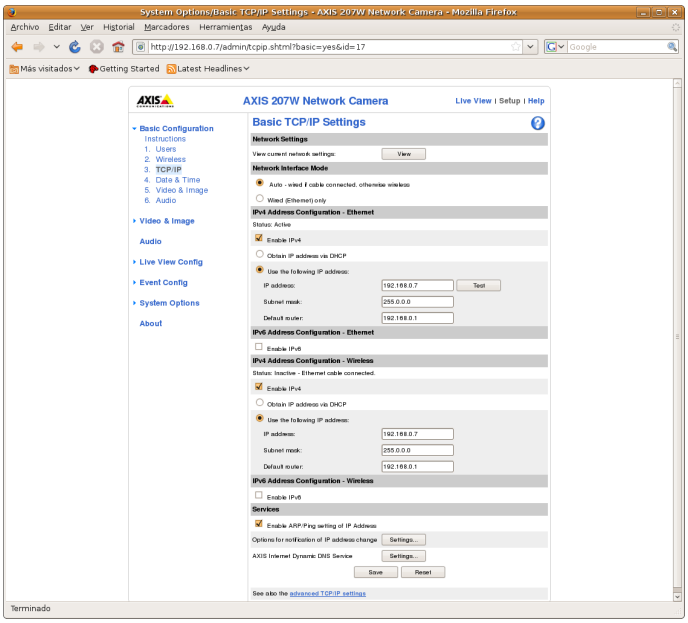

Figura H.3: Cámara IP AXIS 207W. Opciones básica de TCP-IP.

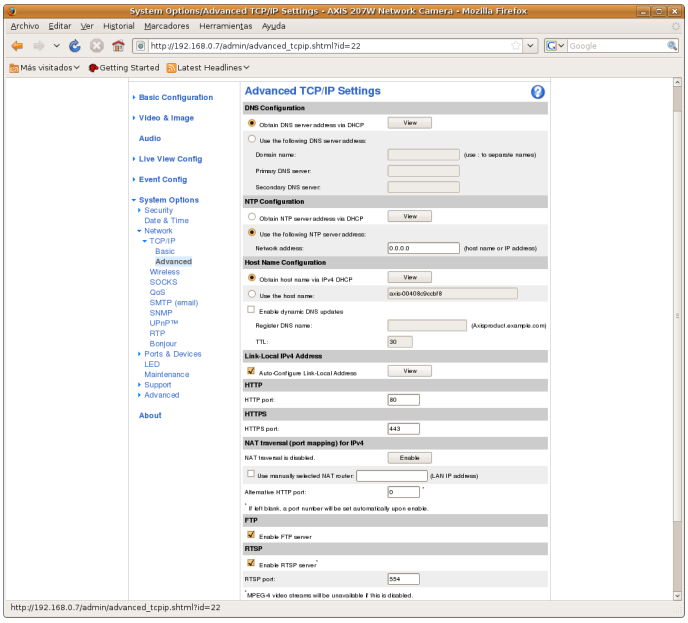

Figura H.4: Cámara IP AXIS 207W. Opciones avanzadas de TCP-IP.

Otras opciones que han sido modificadas de acuerdo a las restricciones existentes en el escenario del proyecto POSEIDON, son el envío a una tasa constante, siendo su valor máximo de 2 Mbps (Figura *H.5*)

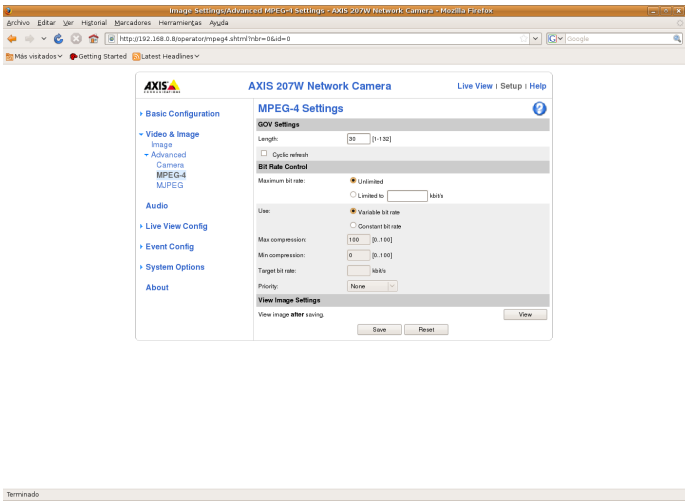

Figura H.5: Cámara IP AXIS 207W. Opciones para MPEG-4.

Por último se muestra en la Figura *H.6* la imagen capturada por la cámara IP.

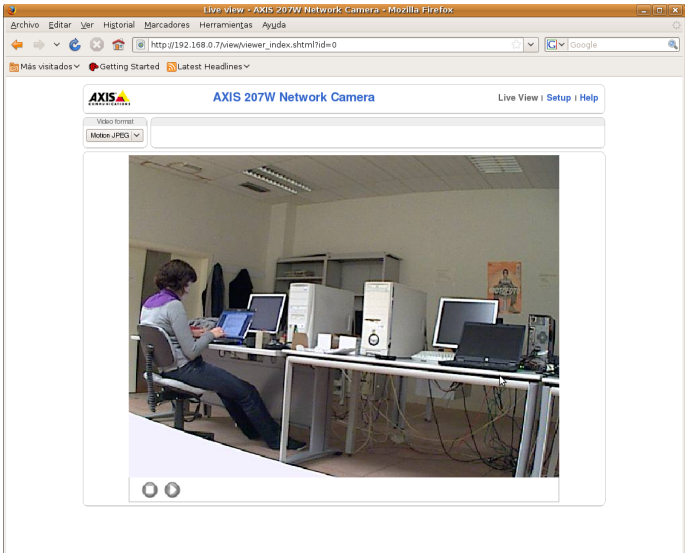

Figura H.6: Cámara IP AXIS 207W. Vista en tiempo real.

Apéndice I

Hardware adicional

I.1. Método de recuperación para un router Fonera

En ocasiones pueden ocurrir ciertos problemas a la hora de instalar el nuevo *firmware* en el router. Por ello cuando la Fonera se quede bloqueada (o *bricked*), la solución es comunicarse con ella a través del puerto serie. En primer lugar se debe utilizar un circuito basado en el chip integrado MAX232, que convierta los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa. La alimentación de este circuito se obtiene directamente del router Fonera. En la Figura *I*.1 se muestra un esquemático de estas conexiones y en la Figura *I*.2 el circuito resultante utilizado para realizar una comunicación con la Fonera a través del puerto serie.

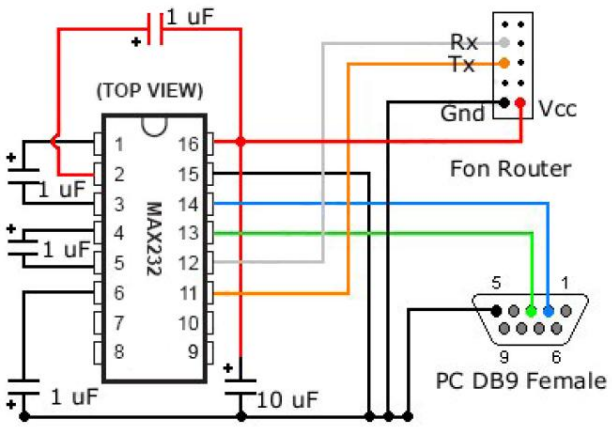

Figura I.1: Esquemático del circuito para la comunicación serie con la Fonera.

Para realizar la comunicación con el router es necesario que el equipo tenga instalado el programa *minicom* (si no se encuentra instalado: `apt-get install minicom`). Se arranca el programa con: `minicom -s` (Figura *I*.3) y se configuran los parámetros del puerto serie (Figura *I*.4).

En primer lugar se debe elegir el dispositivo con el que se va a comunicar el programa, en este caso el router. Para averiguarlo se puede escribir la siguiente instrucción

Figura I.2: Circuito para la comunicación serie con la Fonera.

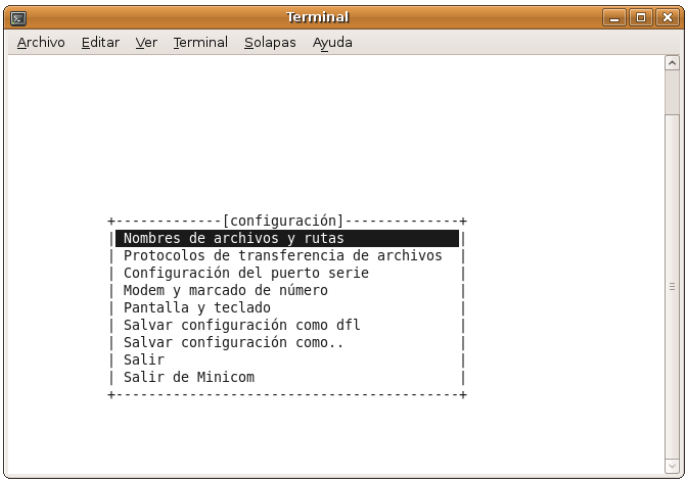

Figura I.3: Ventana de inicio del programa *minicom*.

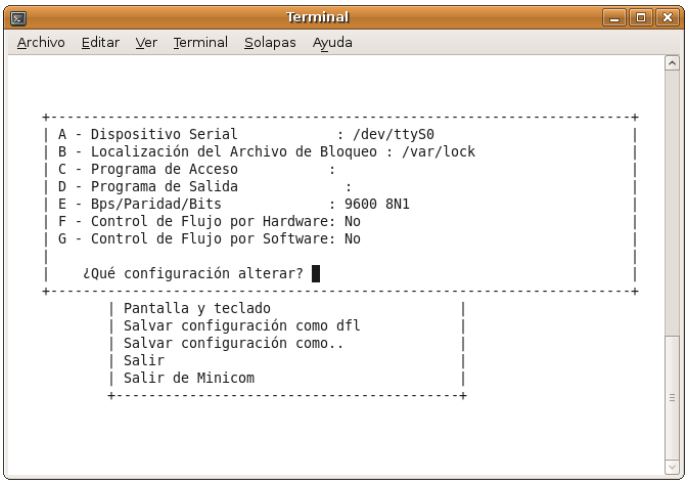

Figura I.4: Parámetros de configuración del puerto serie del programa *minicom*.

teniendo el router conectado `cat /dev/ttyS*`. Normalmente el dispositivo corresponde a la opción ttyS0. Una vez averiguado, se selecciona la opción A y se escribe `/dev/ttyS0`. A continuación se selecciona la opción E para elegir la tasa, la paridad y los datos: `9600 8N1` (Figura *I.5*).

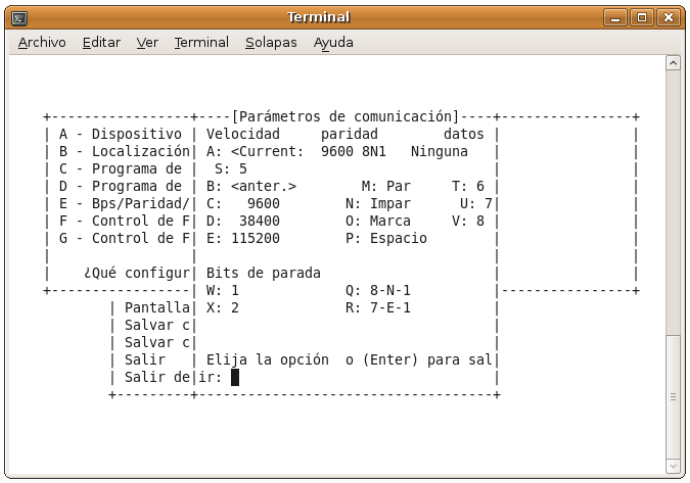

Figura I.5: Parámetros referentes a la comunicación serie en el programa *minicom*.

Por último se guarda la configuración y se elige la opción *Salir* (no *Salir de Minicom*). Ahora se inicia la comunicación con la Fonera, mostrando un mensaje que indica el arranque del kernel, si se presiona intro se carga *RedBoot*, a partir de aquí se pueden seguir de nuevo los pasos explicados en el apartado C.3.4. Si no se hubiese parado el arranque del kernel, habría aparecido un mensaje de error *kernel panic*. Lo que se recomienda es que tan sólo se aplique el comando *fis init* para *flashear* el router y a continuación conectarse a él a través de telnet y seguir los pasos del apartado C.3.4, desestimando en este caso el primero (*fis init*)

Glosario

La lista de los acrónimos utilizados en este proyecto es la siguiente:

AR *Access Router*

AP *Access Point*

BA *Binding ACK*

BR *Binding Refresh*

BU *Binding Update*

CN *Correspondent Node*

CoA *Care-of-Address*

COTS *Commercial Off-The-Shell*

DNS *Domain Name Server*

DHCP *Dynamic Host Configuration Protocol*

HA *Home Agent*

HL *Home Link*

HMIPv6 *Hierarchical MIPv6*

HoA *Home Address*

HTTP *Hypertext Transfer Protocol*

IEEE *Institute of Electrical and Electronics Engineers*

IETF *Internet Engineering Task Force*

IP *Internet Protocol*

IPsec *Internet Protocol Security*

ITU *International Telecommunication Union*

MIP *Mobile IP*

MN *Mobile Node*

MNP *Mobile Network Prefix*

MR *Mobile Router*

NEMO BS *Network Mobility Basic Support*

PMIPv6 *Proxy Mobile IPv6*

RA *Router Advertisement*

RTSP *Real Time Streaming Protocol*

SSH *Secure SHell*

TCP *Transmission Control Protocol*

TELNET *TELetype NETwork*

TFTP *Trivial File Transfer Protocol*

TTL *Time-to-live*

UDP *User Datagram Protocol*

VLAN *Virtual Local Area Network*

WLAN *Wireless Local Area Network*

Bibliografía

[AC07] José Luis Almodóvar Chico, *Implementación de un router móvil para soporte de movilidad de redes IP en un router Linksys WRT54G*, Proyecto fin de carrera, Universidad Carlos III de Madrid . Escuela Politécnica Superior, 2007.

[ADD04] J. Arkko, V. Devarapalli, and F. Dupont, *Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents*, RFC 3776 (Proposed Standard), June 2004, http://www.ietf.org/rfc/rfc3776.txt, Updated by RFC 4877.

[Aur05] T. Aura, *Cryptographically Generated Addresses (CGA)*, RFC 3972 (Proposed Standard), March 2005, http://www.ietf.org/rfc/rfc3972.txt, Updated by RFCs 4581, 4982.

[AVH07] J. Arkko, C. Vogt, and W. Haddad, *Enhanced Route Optimization for Mobile IPv6*, RFC 4866 (Proposed Standard), May 2007, http://www.ietf.org/rfc/rfc4866.txt.

[BA07] Andreas Baldessari, Roberto andFestag and Julien Abeille, *Nemo meets vanet: A deployability analysis of network mobility in vehicular communication*, 7th International Conference on ITS Telecommunications (ITST 2007) (ITST), ITST (Intelligent Tansports Systems Telecommunications), 2007, pp. 375–380.

[BBC04] Carlos J. Bernardos, Marcelo Bagnulo, and María Calderón, *MIRON: MIPv6 Route Optimization for NEMO*, Proceedings of the 4th Workshop on Applications and Services in Wireless Networks (ASWN 2004) (Boston, MA, USA), August 2004, pp. 189–197.

[BGTP07] R. Bonica, D. Gan, D. Tappan, and C. Pignataro, *Extended ICMP to Support Multi-Part Messages*, RFC 4884 (Proposed Standard), April 2007, http://www.ietf.org/rfc/rfc4884.txt.

[BLFM05] T. Berners-Lee, R. Fielding, and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986 (Standard), January 2005, http://www.ietf.org/rfc/rfc3986.txt.

[BOC+06] Carlos J. Bernardos, Antonio De La Oliva, María Calderón, Dirk von Hugo, and Holger Kahle, *NEMO: Network Mobility. Bringing ubiquity to the Internet access*, IEEE INFOCOM 2006, April 2006, demonstration.

[BSC+05a] Carlos J. Bernardos, Ignacio Soto, María Calderón, Dirk von Hugo, and Emmanuel Riou, *NEMO: Movilidad de Redes en IPv6*, Novatica (2005), no. 174, 37–43.

[BSC+05b] Carlos J. Bernardos, Ignacio Soto, María Calderón, Dirk von Hugo, and Emmanuel Riou, *NEMO: Network Mobility in IPv6*, UPGRADE - The European Journal for the Informatics Professional **VI** (2005), no. 2, 36–42.

[BZFL08] Roberto Baldessari, Wenhui Zhang, Andreas Festag, and Long Le, *A manet-centric solution for the application of nemo in vanet using geographic routing*, TridentCom '08: Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities (ICST, Brussels, Belgium, Belgium), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–7.

[car] *Car 2 Car Communication Consortium*, http://www.car-to-car.org/.

[CBB+06] María Calderón, Carlos J. Bernardos, Marcelo Bagnulo, Ignacio Soto, and Antonio de la Oliva, *Design and Experimental Evaluation of a Route Optimization Solution for NEMO*, IEEE Journal on Selected Areas in Communications **24** (2006), no. 9, 1702–1716.

[CD98] A. Conta and S. Deering, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, RFC 2463 (Draft Standard), December 1998, http://www.ietf.org/rfc/rfc2463.txt, Obsoleted by RFC 4443.

[CDG06] A. Conta, S. Deering, and M. Gupta, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, RFC 4443 (Draft Standard), March 2006, http://www.ietf.org/rfc/rfc4443.txt, Updated by RFC 4884.

[CMBB09] Maria Calderon, Hassnaa Moustafa, Carlos J. Bernardos, and Roberto Baldessari, Vehicular Networks: Techniques, Standards and Applications Book, ch. IP Address Autoconfiguration in Vehicular Networks, CRC Press, 2009.

[CN06] S. Chakrabarti and E. Nordmark, *Extension to Sockets API for Mobile IPv6*, RFC 4584 (Informational), July 2006, http://www.ietf.org/rfc/rfc4584.txt.

[DH98] S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 2460 (Draft Standard), December 1998, http://www.ietf.org/rfc/rfc2460.txt, Updated by RFC 5095.

[dlOBC05] Antonio de la Oliva, Carlos J. Bernardos, and María Calderón, *Practical evaluation of a network mobility solution*, Proceedings of the 11th Open European Summer School (EUNICE 2005: Networked Applications) (Colmenarejo, Madrid (SPAIN)), July 2005, pp. 60 – 66.

[DWPT05] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, *Network Mobility (NEMO) Basic Support Protocol*, RFC 3963 (Proposed Standard), January 2005, http://www.ietf.org/rfc/rfc3963.txt.

[Ern07] T. Ernst, *Network Mobility Support Goals and Requirements*, RFC 4886 (Informational), July 2007, http://www.ietf.org/rfc/rfc4886.txt.

[For04] Andrea G. Forte, *Reducing mac layer handoff latency in ieee 802.11 wireless lans*, In MOBIWAC '04: Proceedings of the second international workshop on Mobility management & wireless access protocols, ACM Press, 2004.

[GLD+08] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, *Proxy Mobile IPv6*, RFC 5213 (Proposed Standard), August 2008, http://www.ietf.org/rfc/rfc5213.txt.

[GTB+03] R. Gilligan, S. Thomson, J. Bound, J. McCann, and W. Stevens, *Basic Socket Interface Extensions for IPv6*, RFC 3493 (Informational), February 2003, http://www.ietf.org/rfc/rfc3493.txt.

[GTBS97] R. Gilligan, S. Thomson, J. Bound, and W. Stevens, *Basic Socket Interface Extensions for IPv6*, RFC 2133 (Informational), April 1997, http://www.ietf.org/rfc/rfc2133.txt, Obsoleted by RFC 2553.

[GTBS99] R. Gilligan, S. Thomson, J. Bound, and W. Stevens, *Basic Socket Interface Extensions for IPv6*, RFC 2553 (Informational), March 1999, http://www.ietf.org/rfc/rfc2553.txt, Obsoleted by RFC 3493, updated by RFC 3152.

[Han04] Jun-ichiro itojun Hangino, *Ipv6 network programming*, Digital Press, Newton, MA, USA, 2004.

[Hus05] G. Huston, *Architectural Approaches to Multi-homing for IPv6*, RFC 4177 (Informational), September 2005, http://www.ietf.org/rfc/rfc4177.txt.

[iee07] *IEEE 802.11 Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements*, 2007.

[ITU96] ITU, *ITU-T Rec. P.800, Methods for Subjective Determination of Transmission Quality*, Tech. report, ITU, 1996.

[ITU02] ITU, *Rec. ITU-R BT.500-11. Methodology for the Subjective Assessment of the Quality of Television Pictures*, Tech. report, ITU, 2002.

[JPA04] D. Johnson, C. Perkins, and J. Arkko, *Mobility Support in IPv6*, RFC 3775 (Proposed Standard), June 2004, http://www.ietf.org/rfc/rfc3775.txt.

[Koo05] R. Koodli, *Fast Handovers for Mobile IPv6*, RFC 4068 (Experimental), July 2005, http://www.ietf.org/rfc/rfc4068.txt, Obsoleted by RFC 5268.

[Koo08] R. Koodli, *Mobile IPv6 Fast Handovers*, RFC 5268 (Proposed Standard), June 2008, http://www.ietf.org/rfc/rfc5268.txt.

[McC05] P. McCann, *Mobile IPv6 Fast Handovers for 802.11 Networks*, RFC 4260 (Informational), November 2005, http://www.ietf.org/rfc/rfc4260.txt.

[mob04] *Mobile IPv6 Overview*, Tech. report, Cisco Systems, Diciembre 2004.

[Moo06] N. Moore, *Optimistic Duplicate Address Detection (DAD) for IPv6*, RFC 4429 (Proposed Standard), April 2006, http://www.ietf.org/rfc/rfc4429.txt.

[NCL07] E. Nordmark, S. Chakrabarti, and J. Laganier, *IPv6 Socket API for Source Address Selection*, RFC 5014 (Informational), September 2007, http://www.ietf.org/rfc/rfc5014.txt.

[nem10] *Network Mobility (NEMO)*, Septiembre 2010, http://datatracker.ietf.org/wg/nemo/.

[NEPB07] C. Ng, T. Ernst, E. Paik, and M. Bagnulo, *Analysis of Multihoming in Network Mobility Support*, RFC 4980 (Informational), October 2007, http://www.ietf.org/rfc/rfc4980.txt.

[NNS98] T. Narten, E. Nordmark, and W. Simpson, *Neighbor Discovery for IP Version 6 (IPv6)*, RFC 2461 (Draft Standard), December 1998, http://www.ietf.org/rfc/rfc2461.txt, Obsoleted by RFC 4861, updated by RFC 4311.

[NNSS07] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, *Neighbor Discovery for IP version 6 (IPv6)*, RFC 4861 (Draft Standard), September 2007, http://www.ietf.org/rfc/rfc4861.txt.

[NUO+07] K. Nagami, S. Uda, N. Ogashiwa, H. Esaki, R. Wakikawa, and H. Ohnishi, *Multi-homing for small scale fixed network Using Mobile IP and NEMO*, RFC 4908 (Experimental), June 2007, http://www.ietf.org/rfc/rfc4908.txt.

[OD04] Antonio de la Oliva Delgado, *Estudio de soluciones de movilidad IPv6 e implementación del protocolo NEMO de soporte básico de movilidad de redes*, Proyecto fin de carrera, Universidad Carlos III de Madrid . Escuela Politécnica Superior, Leganés, 2004.

[ope] *OpenWrt Wireless Freedom*, http://openwrt.org/.

[OW09] Stephan Olariu and Michele C. Weigle (eds.), *Vehicular networks: From theory to practice*, CRC Press Taylor & Francis, mar 2009.

[PCB07] C. Perkins, P. Calhoun, and J. Bharatia, *Mobile IPv4 Challenge/Response Extensions (Revised)*, RFC 4721 (Proposed Standard), January 2007, http://www.ietf.org/rfc/rfc4721.txt.

[Per02] C. Perkins, *IP Mobility Support for IPv4*, RFC 3344 (Proposed Standard), August 2002, http://www.ietf.org/rfc/rfc3344.txt, Updated by RFC 4721.

[pos] *Poseidon*, http://enjambre.it.uc3m.es/ poseidon/.

[Ram99] Ramjee Ramachandran, *IP Micro-Mobility Support Using IP Micro-Mobility Support Using HAWAII*, Tech. report, IETF - Internet Engineering Task Force, Julio 1999.

[SBBC09] Ignacio Soto, Roberto Baldessari, Carlos J. Bernardos, and Maria Calderon, Vehicular Networks: Techniques, Standards and Applications Book, ch. Network Mobility (NEMO) in Vehicular Networks, CRC Press, 2009.

[SBC+09a] Ignacio Soto, Carlos J. Bernardos, Maria Calderon, Albert Banchs, and Arturo Azcorra, *NEMO-Enabled Localized Mobility Support for Internet Access in Automotive Scenarios*, IEEE Communications Magazine **47** (2009), no. 5, 152 – 159.

[SBC+09b] Ignacio Soto, Carlos J. Bernardos, María Calderón, Albert Banchs, and Arturo Azcorra, *NEMO-Enabled Localized Mobility Support for Internet Access in Automotive Scenarios*, IEEE Communications Magazine, Special Issue on Automotive Networking - Technology, Design, and Applications series **47** (2009), no. 5, 152–159.

[SCEB08] H. Soliman, C. Castelluccia, K. ElMalki, and L. Bellier, *Hierarchical Mobile IPv6 (HMIPv6) Mobility Management*, RFC 5380 (Proposed Standard), October 2008, http://www.ietf.org/rfc/rfc5380.txt.

[SCMB05] H. Soliman, C. Castelluccia, K. El Malki, and L. Bellier, *Hierarchical Mobile IPv6 Mobility Management (HMIPv6)*, RFC 4140 (Experimental), August 2005, http://www.ietf.org/rfc/rfc4140.txt, Obsoleted by RFC 5380.

[SK08] Ignacio Soto and José Félix Kukielka, *Wireless Local Area Networks: WLANs*, Slides, Universidad Carlos III de Madrid, 2008.

[ST98] W. Stevens and M. Thomas, *Advanced Sockets API for IPv6*, RFC 2292 (Informational), February 1998, http://www.ietf.org/rfc/rfc2292.txt, Obsoleted by RFC 3542.

[Ste90] W. Richard Stevens, *Unix network programming*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[STNJ03] W. Stevens, M. Thomas, E. Nordmark, and T. Jinmei, *Advanced Sockets Application Program Interface (API) for IPv6*, RFC 3542 (Informational), May 2003, http://www.ietf.org/rfc/rfc3542.txt.

[Tan02] Andrew Tanenbaum, *Computer networks*, Prentice Hall Professional Technical Reference, 2002.

[TN98] S. Thomson and T. Narten, *IPv6 Stateless Address Autoconfiguration*, RFC 2462 (Draft Standard), December 1998, http://www.ietf.org/rfc/rfc2462.txt, Obsoleted by RFC 4862.

[TNJ07] S. Thomson, T. Narten, and T. Jinmei, *IPv6 Stateless Address Autoconfiguration*, RFC 4862 (Draft Standard), September 2007, http://www.ietf.org/rfc/rfc4862.txt.

[TWD07] P. Thubert, R. Wakikawa, and V. Devarapalli, *Network Mobility Home Network Models*, RFC 4887 (Informational), July 2007, http://www.ietf.org/rfc/rfc4887.txt.

[veh] *National Highway Traffic Safety Administration*, http://www.nhtsa.gov/.

[Zul] Holger Zuleger, *Mobile Internet Protocol v6. MIPv6. A short introduction*, Tech. report.